LoadRunner 使用手册

测试中心　　刘艳会

1　LoadRunner 概要介绍

LoadRunner® 是一种预测系统行为和性能的工业标准级负载测试工具。通过以模拟上千万用户实施并发负载及实时性能监测的方式来确认和查找问题，LoadRunner 能够对整个企业架构进行测试。通过使用 LoadRunner ，企业能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。

目前企业的网络应用环境都必须支持大量用户，网络体系架构中含各类应用环境且由不同供应商提供软件和硬件产品。难以预知的用户负载和愈来愈复杂的应用环境使公司时时担心会发生用户响应速度过慢，系统崩溃等问题。这些都不可避免地导致公司收益的损失。Mercury Interactive 的 LoadRunner 能让企业保护自己的收入来源，无需购置额外硬件而最大限度地利用现有的 IT 资源，并确保终端用户在应用系统的各个环节中对其测试应用的质量，可靠性和可扩展性都有良好的评价。

LoadRunner 是一种适用于各种体系架构的自动负载测试工具，它能预测系统行为并优化系统性能。LoadRunner 的测试对象是整个企业的系统，它通过模拟实际用户的操作行为和实行实时性能监测，来帮助您更快的查找和发现问题。此外，LoadRunner 能支持广范的协议和技术，为您的特殊环境提供特殊的解决方案。

1.1 轻松创建虚拟用户

使用 LoadRunner 的 Virtual User Generator，您能很简便地创立起系统负载。该引擎能够生成虚拟用户，以虚拟用户的方式模拟真实用户的业务操作行为。它先记录下业务流程(如下订单或机票预定)，然后将其转化为测试脚本。利用虚拟用户，您可以在 Windows ，UNIX 或 Linux 机器上同时产生成千上万个用户访问。所以 LoadRunner 能极大的减少负载测试所需的硬件和人力资源。另外，LoadRunner 的 TurboLoad 专利技术能提供很高的适应性。TurboLoad 使您可以产生每天几十万名在线用户和数以百万计的点击数的负载。

用 Virtual User Generator 建立测试脚本后，您可以对其进行参数化操作，这一操作能让您利用几套不同的实际发生数据来测试您的应用程序，从而反映出本系统的负载能力。以一个订单输入过程为例，参数化操作可将记录中的固定数据，如订单号和客户名称，由可变值来代替。在这些变量内随意输入可能的订单号和客户名，来匹配多个实际用户的操作行为。

LoadRunner 通过它的 Data Wizard 来自动实现其测试数据的参数化。Data Wizard 直接

连于数据库服务器，从中您可以获取所需的数据（如定单号和用户名）并直接将其输入到测试脚本。这样避免了人工处理数据的需要，Data Wizard 为您节省了大量的时间。

为了进一步确定您的 Virtual user 能够模拟真实用户，您可利用 LoadRunner 控制某些行为特性。例如，只需要点击一下鼠标，您就能轻易控制交易的数量，交易频率，用户的思考时间和连接速度等。

1.2 创建真实的负载

Virtual users 建立起后，您需要设定您的负载方案，业务流程组合和虚拟用户数量。用 LoadRunner 的 Controller，您能很快组织起多用户的测试方案。Controller 的 Rendezvous 功能提供一个互动的环境，在其中您既能建立起持续且循环的负载，又能管理和驱动负载测试方案。而且，您可以利用它的日程计划服务来定义用户在什么时候访问系统以产生负载。这样，您就能将测试过程自动化。同样您还可以用 Controller 来限定您的负载方案，在这个方案中所有的用户同时执行一个动作---如登陆到一个库存应用程序----来模拟峰值负载的情况。另外，您还能监测系统架构中各个组件的性能---- 包括服务器，数据库，网络设备等----来帮助客户决定系统的配置。

LoadRunner 通过它的 AutoLoad 技术，为您提供更多的测试灵活性。使用 AutoLoad ，您可以根据目前的用户人数事先设定测试目标，优化测试流程。例如，您的目标可以是确定您的应用系统承受的每秒点击数或每秒的交易量。

1.3 实时监测器

LoadRunner 内含集成的实时监测器，在负载测试过程的任何时候，您都可以观察到应用系统的运行性能。这些性能监测器为您实时显示交易性能数据（如响应时间）和其它系统组件包括 application server, web server，网路设备和数据库等的实时性能。这样，您就可以在测试过程中从客户和服务器的双方面评估这些系统组件的运行性能，从而更快地发现问题。

再者，利用 LoadRunner 的 ContentCheck TM ，您可以判断负载下的应用程序功能正常与否。ContentCheck 在 Virtual users 运行时，检测应用程序的网络数据包内容，从中确定是否有错误内容传送出去。它的实时浏览器帮助您从终端用户角度观察程序性能状况。

1.4 分析结果以精确定位问题所在

一旦测试完毕后，LoadRunner 收集汇总所有的测试数据，并为您提供高级的分析和报告工具，以便迅速查找到性能问题并追溯原由。使用 LoadRunner 的 Web 交易细节监测器，您可以了解到将所有的图象、框架和文本下载到每一网页上所需的时间。例如，这个交易细节分析机制能够分析是否因为一个大尺寸的图形文件或是第三方的数据组件造成应用系统运行速度减慢。另外，Web 交易细节监测器分解用于客户端、网络和服务器上端到端的反

应时间，便于确认问题，定位查找真正出错的组件。例如，您可以将网络延时进行分解，以判断 DNS 解析时间，连接服务器或 SSL 认证所花费的时间。通过使用 LoadRunner 的分析工具，您能很快地查找到出错的位置和原因并作出相应的调整。

1.5 重复测试保证系统发布的高性能

负载测试是一个重复过程。每次处理完一个出错情况，您都需要对您的应用程序在相同的方案下，再进行一次负载测试。以此检验您所做的修正是否改善了运行性能。

1.6 其他特性

利用 LoadRunner, 您可以很方便地了解系统的性能。 它的 Controller 允许您重复执行与出错修改前相同的测试方案。它的基于 HTML 的报告为您提供一个比较性能结果所需的基准，以此衡量在一段时间内，有多大程度的改进并确保应用成功。由于这些报告是基于 HTML 的文本，您可以将其公布于您公司的内部网上，便于随时查阅。

所有 Mercury Interactive 的产品和服务都是集成设计的, 能完全相容地一起运作。由于它们具有相同的核心技术，来自于 LoadRunner 和 ActiveTest TM 的测试脚本，在 Mercury Interactive 的负载测试服务项目中，可以被重复用于性能监测。借助 Mercury Interactive 的监测功能－－Topaz TM 和 ActiveWatch TM ，测试脚本可重复使用从而平衡投资收益。更重要的是，您能为测试的前期布署和生产系统的监测提供一个完整的应用性能管理解决方案。

- Enterprise Java Beans 的测试

LoadRunner 完全支持 EJB 的负载测试。这些基于 Java 的组件运行在应用服务器上，提供广泛的应用服务。通过测试这些组件，您可以在应用程序开发的早期就确认并解决可能产生的问题。

- 支持无线应用协议

随着无线设备数量和种类的增多，您的测试计划需要同时满足传统的基于浏览器的用户和无线互联网设备，如手机和 PDA。LoadRunner 支持 2 项最广泛使用的协议：WAP 和 I-mode。此外，通过负载测试系统整体架构，LoadRunner 能让您只需要通过记录一次脚本，就可完全检测上述这些无线互联网系统。

- 支持 Media Stream 应用

LoadRunner 还能支持 Media Stream 应用。为了保证终端用户得到良好的操作体验和高质量 Media Stream，您需要检测您的 Media Stream 应用程序。使用 LoadRunner ，您可以记录和重放任何流行的多媒体数据流格式来诊断系统的性能问题，查找原由，分析数据的质量。

- 完整的企业应用环境的支持

LoadRunner 支持广泛的协议，可以测试各种 IT 基础架构。

2 安装 LoadRunner

LoadRunner 分为 Windows 版本和 Unix 版本。如果我们的所有测试环境基于 Windows 平台，那么我们只要安装 Windows 版本即可。

本章讲解的安装过程就是 LoadRunner7.51 的 Windows 版本的安装。

2.1 系统要求

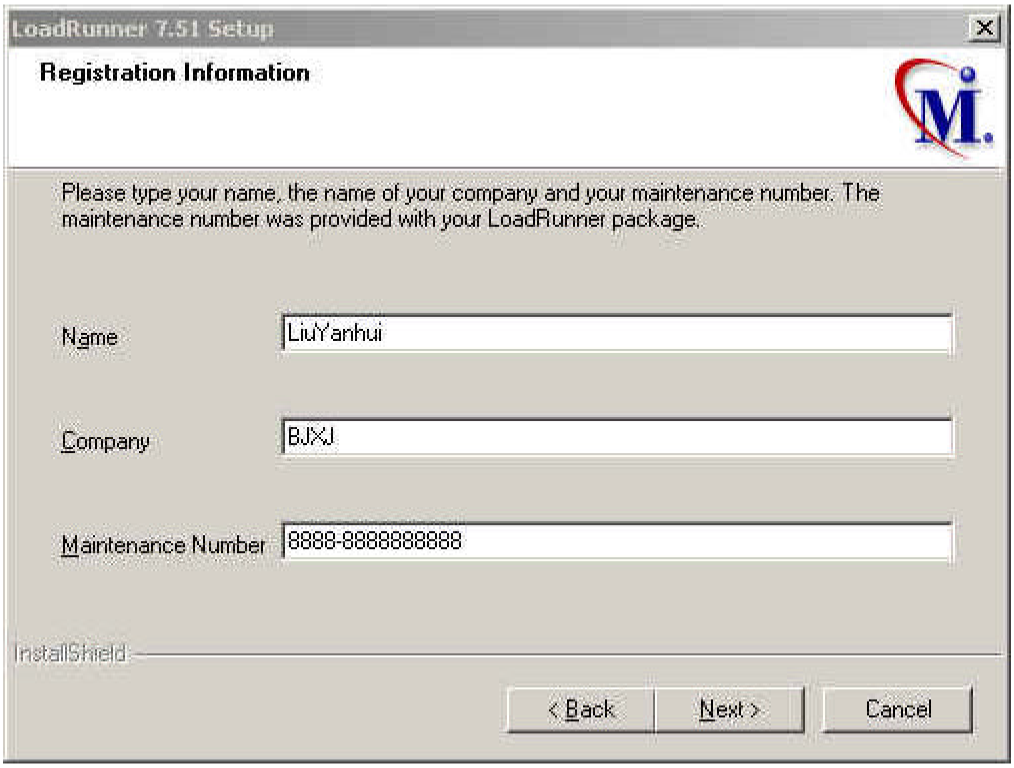
目前部门的测试机和工作机器足可以满足 LoadRunner7.51 的最低要求。不过要比较好的运行 LoadRunner，内存最好在 128M 以上，安装 LoadRunner 的磁盘空间至少剩余 500M。操作系统最好为 Windows 2000。

2.2 安装过程

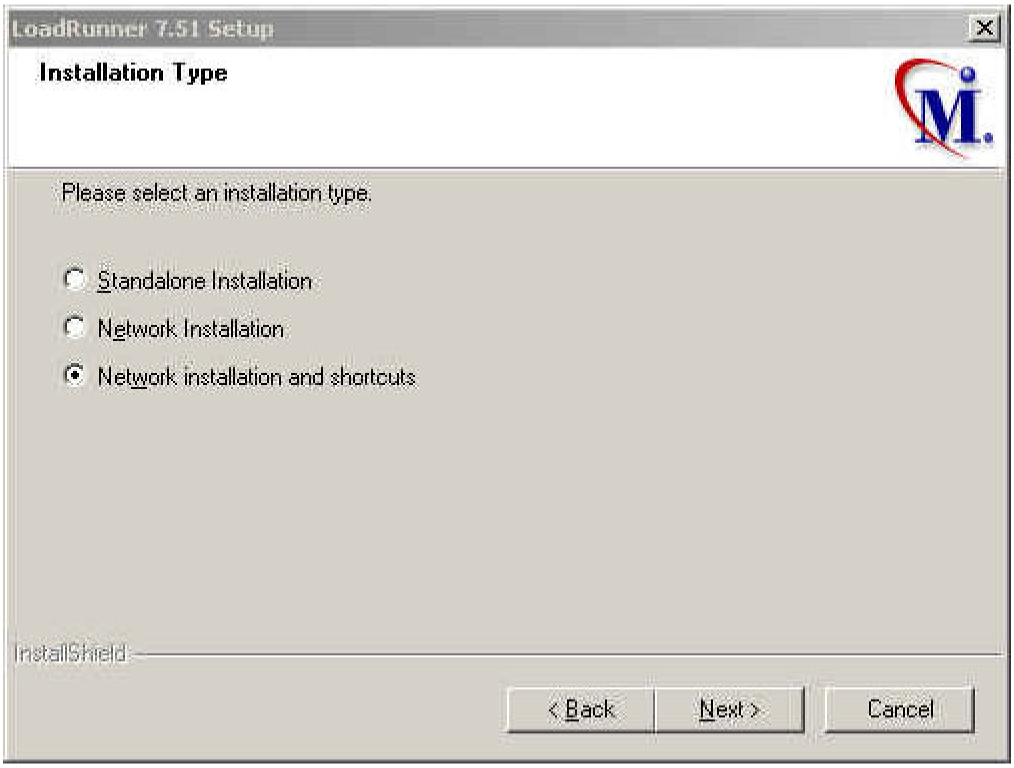
LoadRunner 的安装过程比较简单，这里我仅作简单的说明。

要开始安装 LoadRunner ,以 Administrator 的身份登陆 Windows2000 后,运行 LoadRunner 安装目录中 win32 下 Setup.exe 即可进入安装程序。

1. 在“Registration Information”界面中，输入序列号（不用改动，就是 n 个 8）



2. 在安装类型界面中，选择一种安装类型

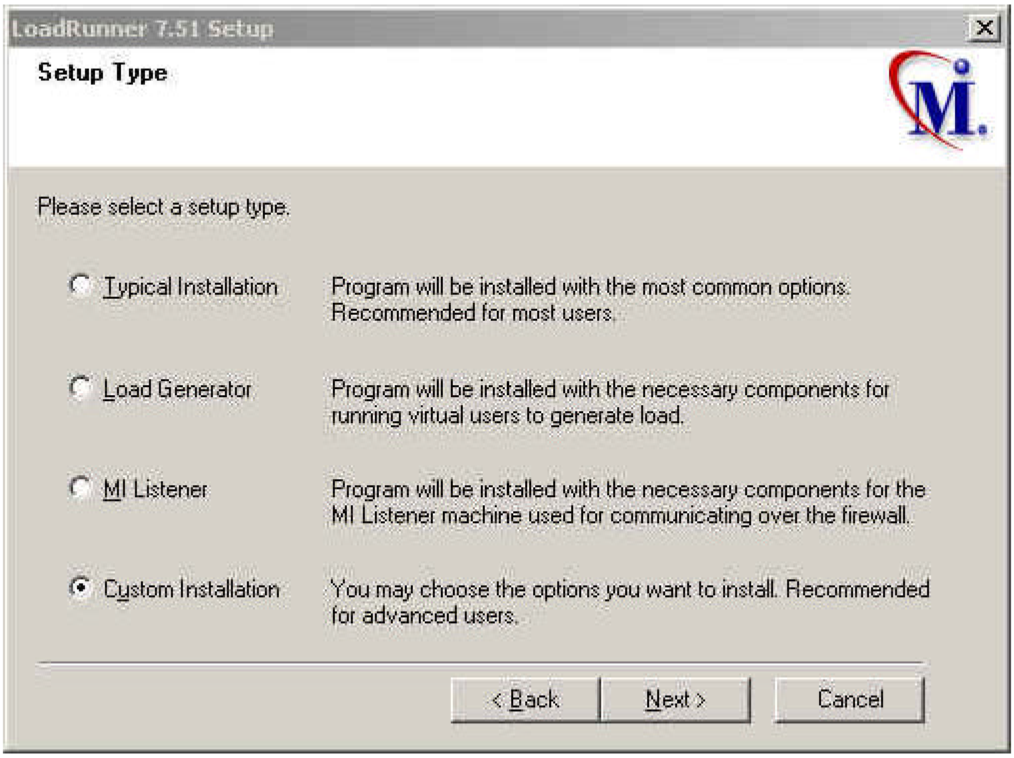


下面简单的对这三种安装类型进行介绍

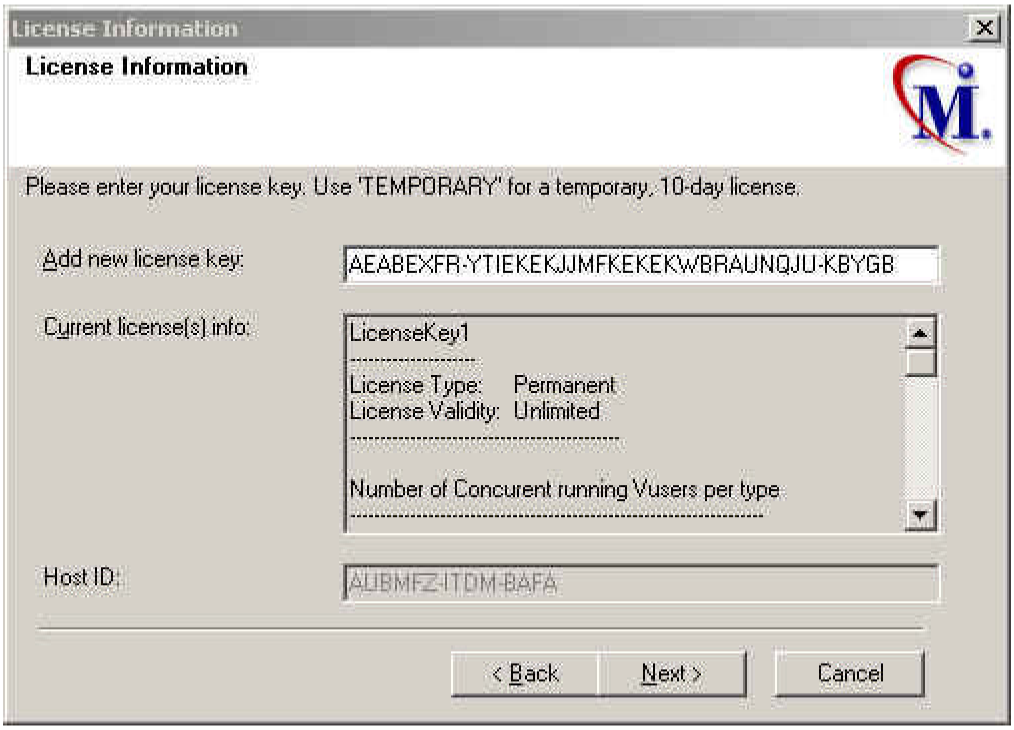
- Standalone Installation　**将要安装** LoadRunner **在一台计算机上**
- Network Installation　**把** LoadRunner **安装在一个网络驱动器上，这样任何能连接到这个网络驱动器的计算机都可以使用** LoadRunner **的部分或者全部组件。**
- Network Installation and shortcuts　**和** Network Installation **类似，不同的只是这种类型将把自己的计算机配置成** Workstation **来运行** LoadRunner**。如果选择了第二项，我们还需要进行** 2.3 **的安装来配置** Workstation.

考虑到我们部门现在的状况，我认为应该选择第三种安装方法。如果只是自己学习研究，选择第一种安装方法。

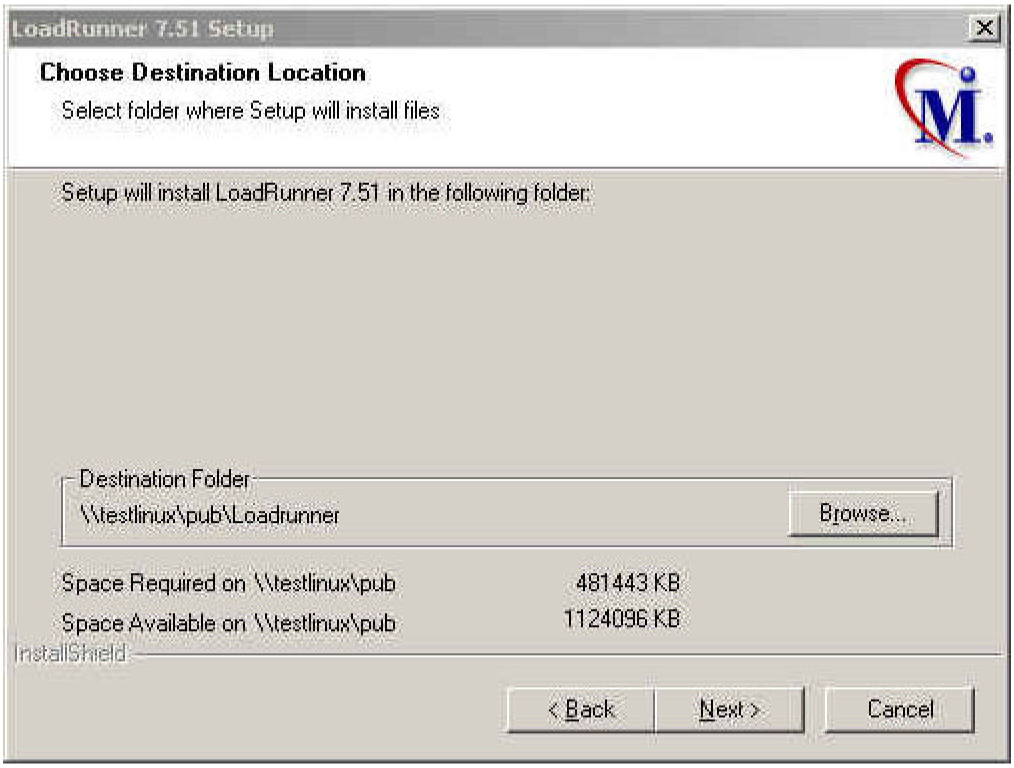
3. **在安装方式界面中，需要选择一种安装方式。建议选择"自定义安装"，这样所有的组件都会一次安装。**
   下面简单的对各个安装方式进行介绍
   - Typical Installation　**安装比较通用的组件，包括** Controller、Vuser、**在线帮助和脚本例程。该选项适合于控制** Vusers **的机器。**
   - Load Generator　**只安装运行** Vusers **产生负载的组件。该选项适合于只产生负载，而不控制** Vusers **的机器。**
   - MI Listener　**安装** MI Listener **组件，用来透过防火墙来运行** Vusers **并且监视性能。**
   - Custom Installation　**自定义安装，我们将使用该选项，安装全部的组件。**



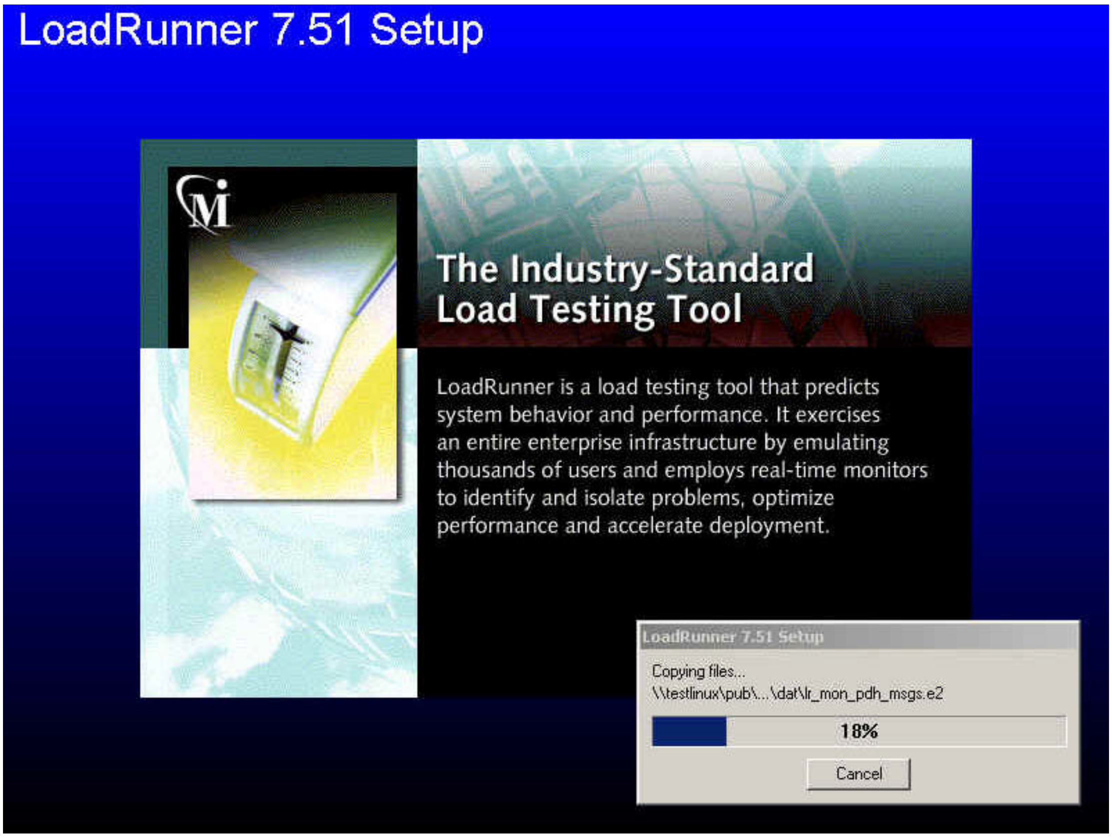
4.　在“License Information”中输入 License Key 后，Next，继续



5.　安装路径选择界面　既然是网络安装，当然要安装到一个网络驱动器上。提醒：最好把网络驱动器映射成本机的一个盘符（比如 H:盘），安装 LoadRunner 的各级目录不要包含中文字符。



6.　Next 后进入拷贝文件的界面

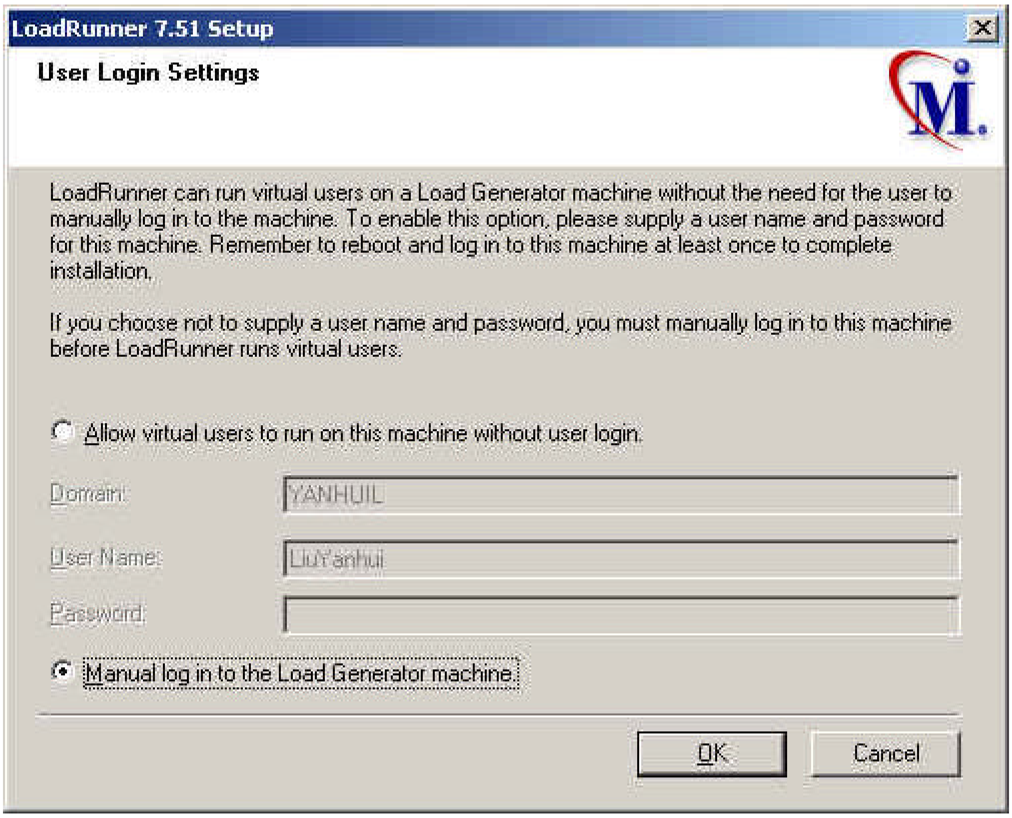


7.　拷贝文件完成后，进入“User Login Settings”界面。

- Allow virtual users to run on this machine without user login 需要在下面输入域、用

户名和密码，这样运行 Load Generator 的机器会自动登陆到网络，

- Manual log in to the Load Generator machine 运行 Vusers 时，自动登陆到网络，无需登陆用户名和密码，这样 Vusers 就会不用任何干预自动的启动运行。推荐选择该项。



这里选择第一项和第二项都可以。

8. 重新启动，输入映射网络驱动器的密码后，安装完成

2.3 WorkStation 安装

LoadRunner 安装在网络驱动器上后，其他的计算机可以访问网络驱动器，安装 Workstation，这样大家就可以通过网络使用 LoadRunner 的共享版本。

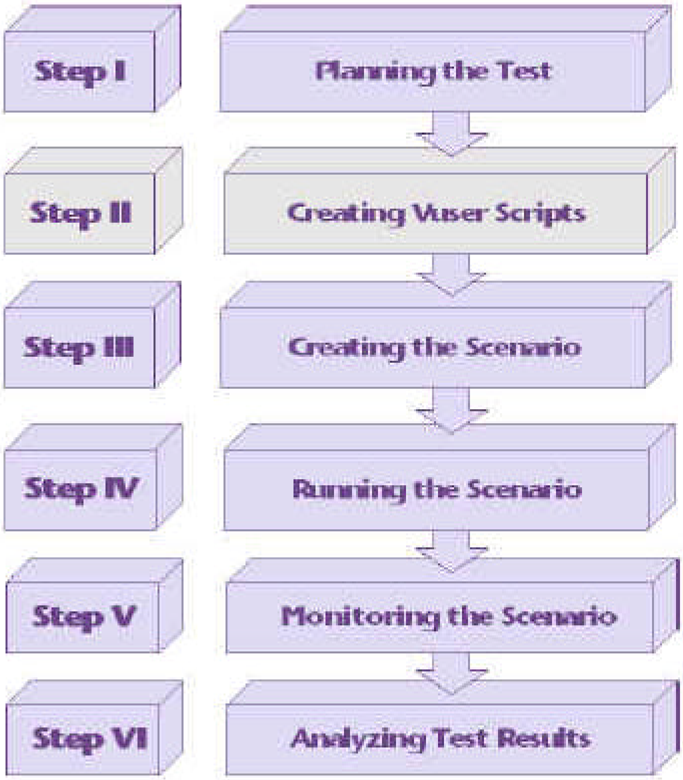
安装过程的主要步骤如下：

1. 把网络驱动器映射到本机的一个盘符。
2. 打开网络驱动器，运行其中的 Setup 目录下的 Setup.exe
3. 剩下的步骤可以参考 2.2 中的步骤了。

3 使用 LoadRunner 对 Web 应用进行负载/压力测试

LoadRunner 包含很多组件，其中最常用的有 Visual User Generator（以下简称 VuGen）、Controller，Analysis。

使用 LoadRunner 进行测试的过程可以用下图表示



摘自 LoadRunner 在线帮助

下面我们就按照上图的步骤来简单说明使用 LoadRunner 的测试过程。

3.1 制定负载测试计划

在任何类型的测试中，测试计划都是必要的步骤。测试计划是进行成功的负载测试的关键。任何类型的测试的第一步都是制定比较详细的测试计划。一个比较好的测试计划能够保证 LoadRunner 能够完成负载测试的目标。

关于比较详细的信息请参考第 4 章。

3.2 开发测试脚本

LoadRunner 使用虚拟用户的活动来模拟真实用户来操作 Web 应用程序，而虚拟用户的活动就包含在测试脚本中，所以说测试脚本对于测试来说是非常重要的。

开发测试脚本要使用 VuGen 组件。测试脚本要完成的内容有：

- 每一个虚拟用户的活动
- 定义结合点
- 定义事务

关于比较详细的信息请参考第 5 章。

3.3 创建运行场景

运行场景描述在测试活动中发生的各种事件。一个运行场景包括一个运行虚拟用户活动的 Load Generator 机器列表，一个测试脚本的列表以及大量的虚拟用户和虚拟用户组。

我们使用 Controller 来创建运行场景。关于比较详细的信息请参考第 6 章。

3.4 运行测试

一切配置妥当，开始运行测试。无需多讲！

3.5 监视场景

在运行过程中，可以监视各个服务器的运行情况（DataBase Server、Web Server 等）。监视场景通过添加性能计数器来实现。

关于比较详细的信息请参考第 7 章。

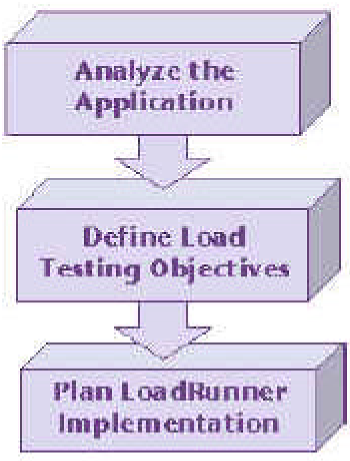
3.6 分析测试结果

所有前面的准备都是为了这一步。我们需要分析大量的图表，生成各种不同的报告，最后会得出结论。

关于比较详细的信息请参考第 8 章。

4 制定负载测试计划

制定负载测试计划一般情况下需要三个步骤，可以用下图表示



摘自 LoadRunner 在线帮助

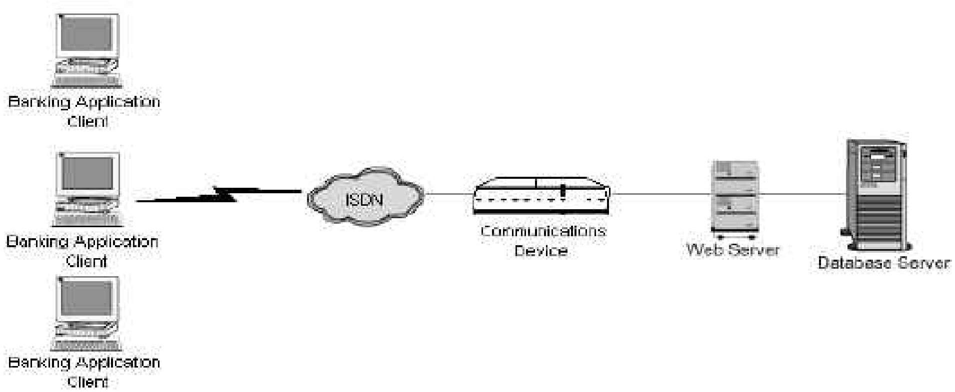
下面简单的对这三个过程进行介绍。

4.1 分析应用程序（Analyze the Application）

制定负载测试计划的第一步是分析应用程序。你应该对系统的软硬件以及配置情况非常的熟悉，这样才能保证你使用 LoadRunner 创建的测试环境真实的反映实际运行的环境。

- **确定系统的组成**

  画出系统的组成图。组成图要包括系统中所有的组件，以及相互之间是如何通讯的。下面是一个系统组成图的例子，可以参考。



- **描述系统配置**

  画出系统组成图后，试着回答以下问题，对组成图进行完善。
  - ✓ 预计有多少用户会连到系统
  - ✓ 客户机的配置情况（硬件、内存、操作系统、软件工具等）
  - ✓ 服务器使用什么类型的数据库以及服务器的配置情况
  - ✓ 客户机和服务器之间如何通讯
  - ✓ 还有什么组件会影响 Response Time 指标（比如 Modem 等）
  - ✓ 通讯装置（网卡、路由器等）的吞吐量是多少？每个通讯装置能够处理多少并发用户
- **分析最普遍的使用方法**

  了解该系统最常用的功能，确定那些功能需要优先测试、什么角色使用该系统以及每个角色会有多少人、每个角色的地理分布情况等，从而预测负载的最高峰出现的情况。

4.2 确定测试目标（Defining Testing Objectives）

这里借用一段文字来说明如何确定测试目标。

Objective	Answers the Question
Measuring end-user response time	How long does it take to complete a business process?
Defining optimal hardware configuration	Which hardware configuration provides the best performance?
Checking reliability	How hard or long can the system work without errors or failures?
Checking hardware or software upgrades	How does the upgrade affect performance or reliability?
Evaluating new products	Which server hardware or software should you choose?
Measuring system capacity	How much load can the system handle without significant performance degradation?
Identifying bottlenecks	Which element is slowing down response time?

摘自 LoadRunner 在线帮助

在这里还要确定何时开始负载测试，在不同的阶段进行什么内容的负载测试。可以用下表来说明。

Planning and Design	Development	Deployment	Production	Evolution
Evaluate new products	Measure response time	Check reliability	Measure response time	Check HW or SW upgrades
Measure response time	Check optimal hardware configuration	Measure response time	Identify bottlenecks	Measure system capacity
	Check HW or SW upgrades	Measure system capacity		
	Check reliability			

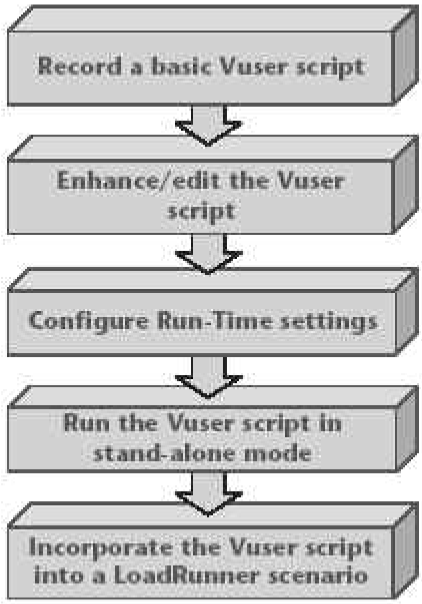
4.3 计划怎样执行 LoadRunner

确定要使用 LoadRunner 度量那些性能参数，根据测量结果计算那些参数，从而可以确定 Vusers（虚拟用户）的活动，最终可以确定那些是系统的瓶颈等。

在这里还要选择测试环境，测试机器的配置情况等等。

5 开发测试脚本

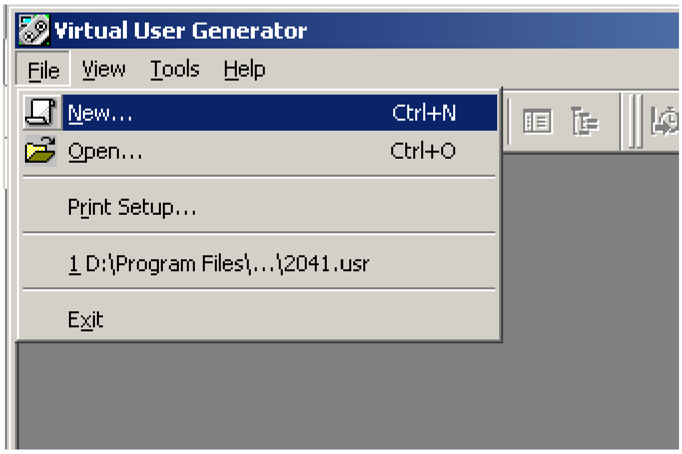
开发测试脚本需要几个步骤，可以用下图来表示



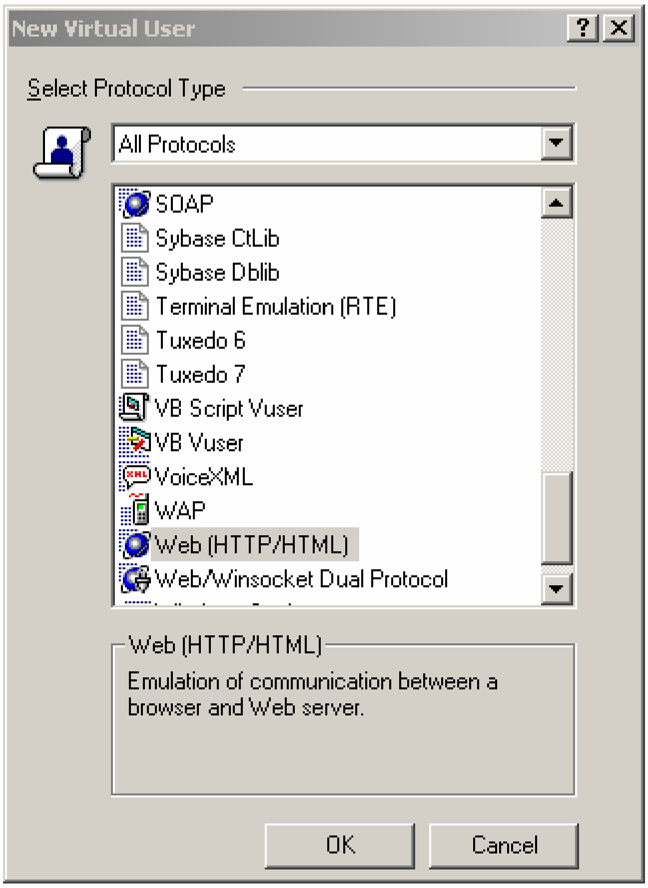
创建用户脚本需要用到 VuGen。**提示：运行 VuGen 最好在 1024*768 的分辨率下，否则有些工具栏会看不到。**

5.1 录制基本的用户脚本

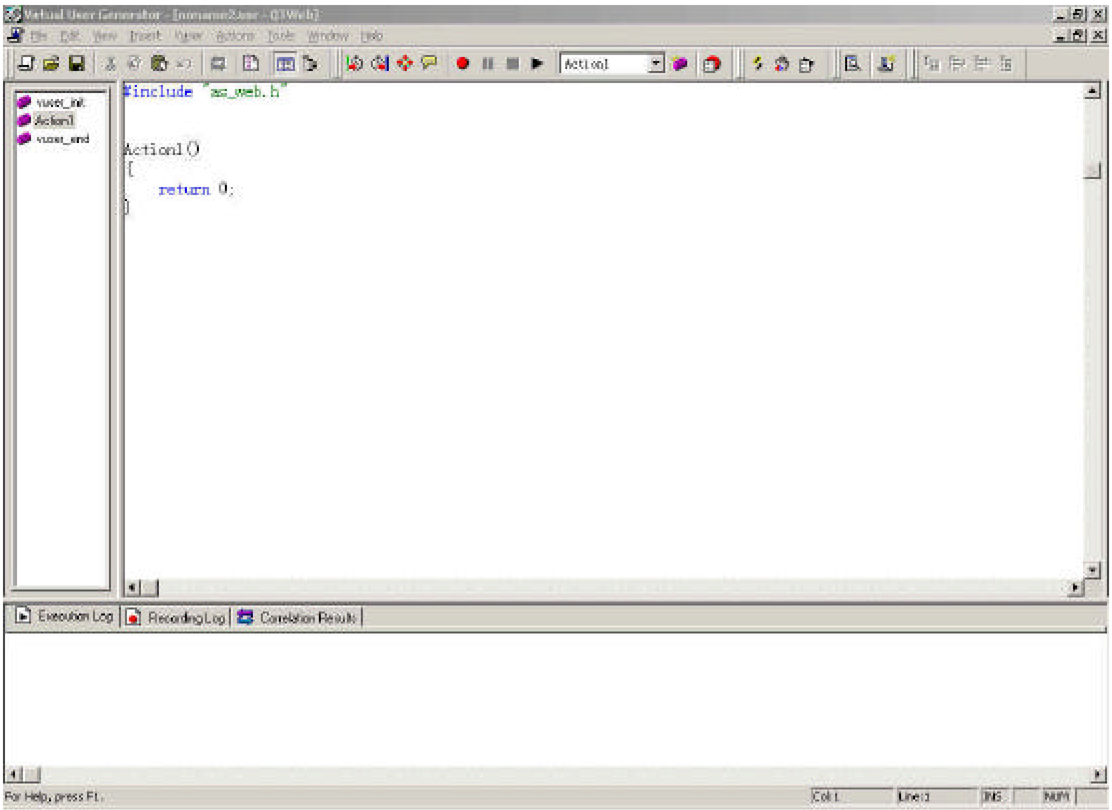
启动 Visual User Generator **后，通过菜单**



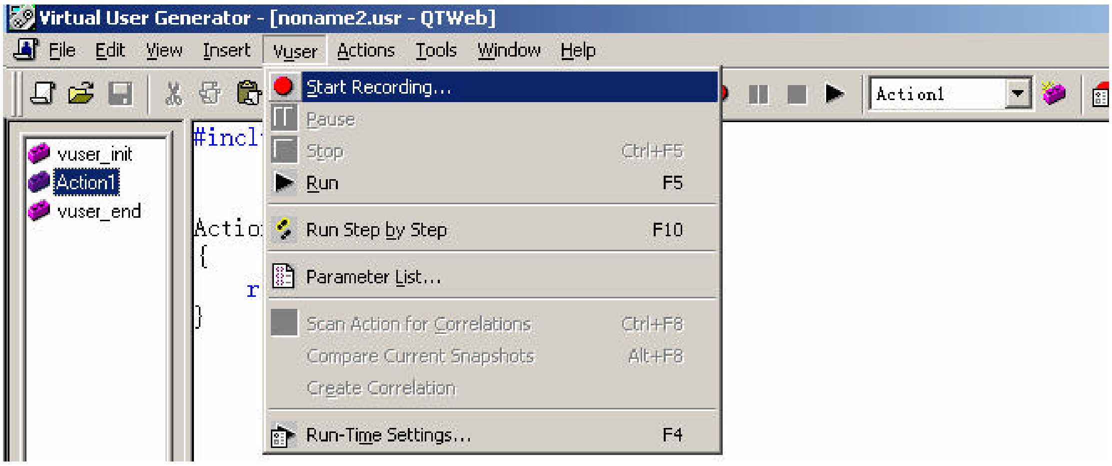
新建一个用户脚本，选择系统通讯的协议。



这里我们需要测试的是 Web 应用，所以我们需要选择 Web(HTTP/HTML)协议，确定后，进入主窗体。

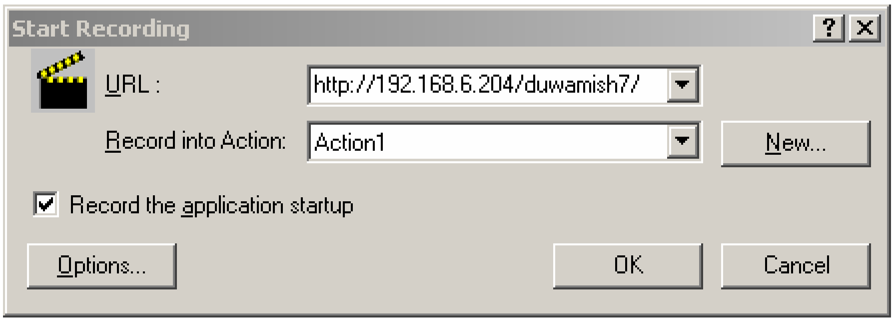


通过菜单



来启动录制脚本的命令。

- ◆ **在** URL **中添入要测试的** Web **站点地址，这里我们以著名的** Duwamish **应用为例子来进行录制。**
- ◆ **选择要把录制的脚本放到哪一个部分，默认情况下是“** Action1 **”。**



这里简单说明一下：VuGen **中的脚本分为三部分：**vuser_init、vuser_end **和** Action。**其中** vuser_init **和** vuser_end **都只能存在一个，不能再分割，而** Action **还可以分成无数多个部分（通过点击** New **按钮，新建** ActionXXX)。

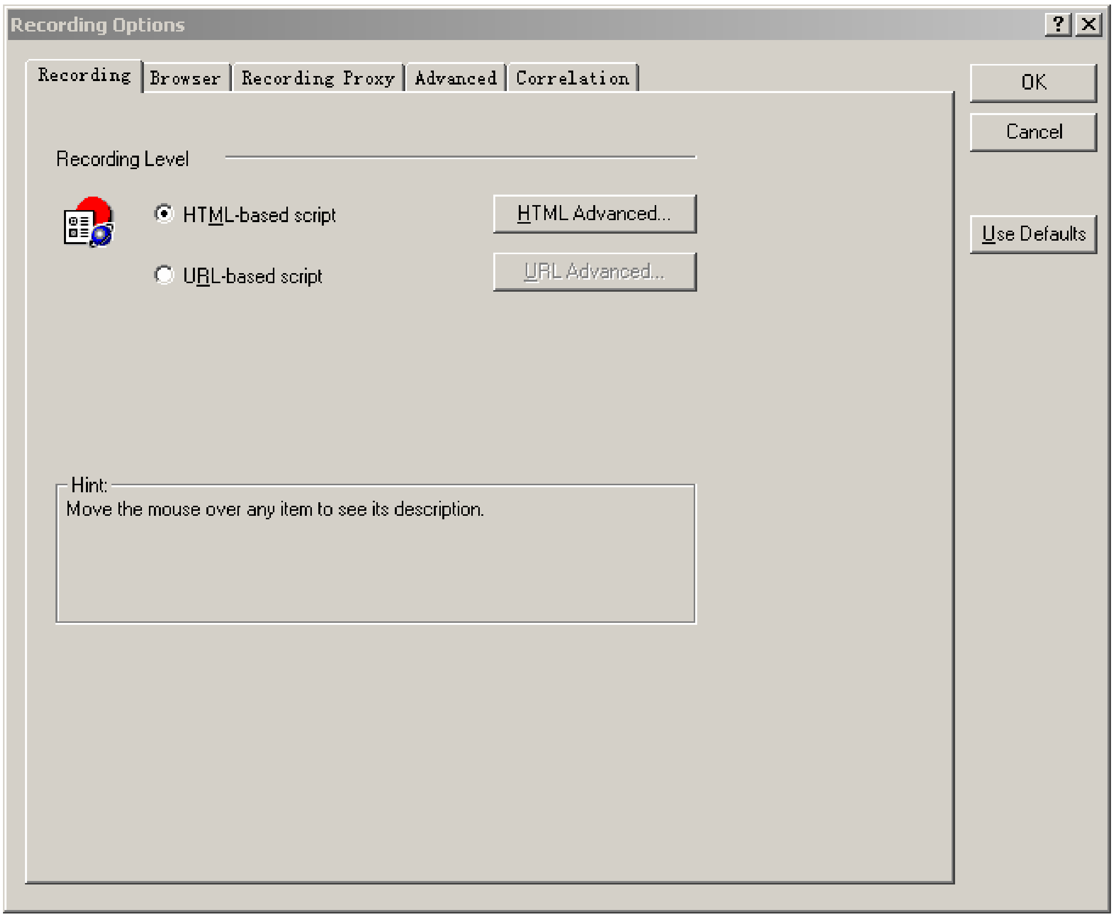
在录制需要登陆的系统时，我们把登陆部分放到 vuser_init **中，把登陆后的操作部分放到** Action **中，把注销关闭登陆部分放到** vuser_end **中。*（如果需要在登陆操作设集合点，那么登陆操作也要放到 Action 中，因为 vuser_init 中不能添加集合点）*在其他情况下，我们只要把操作部分放到** Action **中即可。**

注意：在重复执行测试脚本时，vuser_init 和 vuser_end 中的内容只会执行一次，重复执行的只是 Action 中的部分。

- ◆ “ Record the application startup ”**默认情况下是选中的，说明应用程序一旦启动，**VuGen **就会开始录制脚本；如果没有选中，应用程序启动后，**VuGen **出现以下对话框，并且暂时不会开始录制脚本，用户操作应用程序到需要录制的地方，按下“** Record **”按钮，**VuGen **才开始录制。**



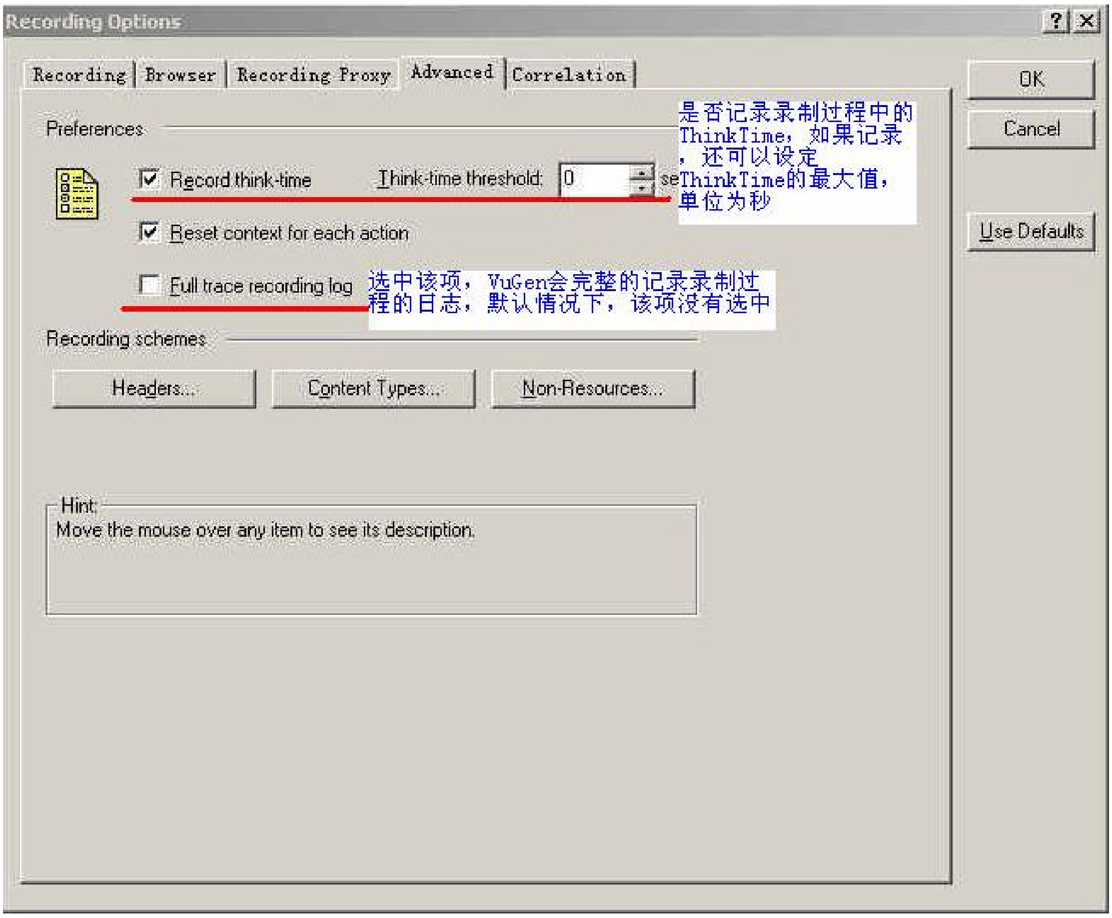
◆ 点“Options”按钮，进入录制的设置窗体，这里一般情况下不需要改动。



- Recording 标签页：默认情况下选择“HTML-based Script”，说明脚本中采用 HTML 页面的形式来表示，这种方式的 Script 脚本容易维护，容易理解，推荐这种方式录制（微软在 ACT 中附带的 Duwamish7 例子采用的就是这种方式）。“URL-based Script”说明脚本中的表示采用基于 URL 的方式，WAS 和 ACT 中的录制方式就是这种，这种方式看上去比较乱。
  选择哪种方式录制，有以下参考原则：
  1 基于浏览器的应用程序推荐使用 HTML-based Script
  2 不是基于浏览器的应用程序推荐使用 URL-based Script。
  3 如果基于浏览器的应用程序中包含了 JavaScript 并且该脚本向服务器产生了请求，比如 DataGrid 的分页按钮等，也要使用 URL-based 方式录制
  4 基于浏览器的应用程序中使用了 HTTPS 安全协议，使用 URL-based 方式录制
- Advanced 标签页：取默认情况即可。下面的图简单的说明了各项的含义。
- Correlation 标签页：这里的内容比较重要，需要定制，主要是为了在录制过程中设置自动关联。这里资料比较少，现在还没有进行深入的研究。

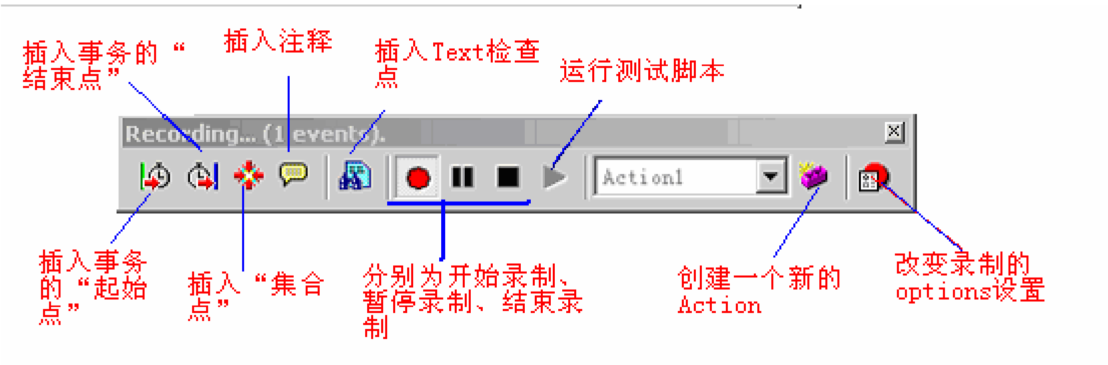
由于内容比较多，就不再一一介绍各项的含义了。

根据自己的需求，选择适当的设置，然后点“OK”后，VuGen 开始录制脚本。
在录制过程中，不要使用浏览器的“后退”功能，LoadRunner 支持不太好！



录制过程中，在屏幕上会有一个工具条出现。

下面我们简单介绍一下各个按钮的功能。



录制的过程和 WinRunner **有些类似，不再多介绍。**

录制完成后，按下“结束录制”按钮，VuGen **自动生成用户脚本，退出录制过程。脚本参考下页的图。**



5.2 完善测试脚本

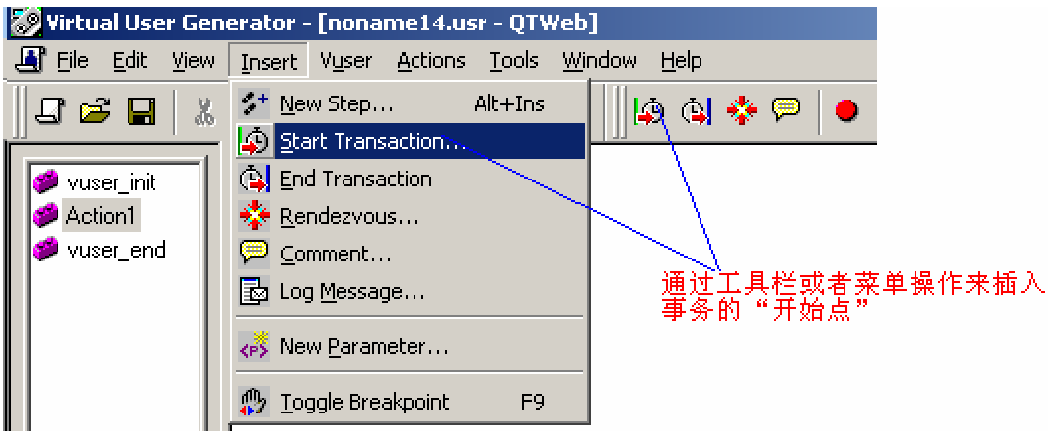
当录制完一个基本的用户脚本后，在正式使用前我们还需要完善测试脚本，增强脚本的灵活性。一般情况下，我们通过以下方法来完善测试脚本。

5.2.1 插入事务

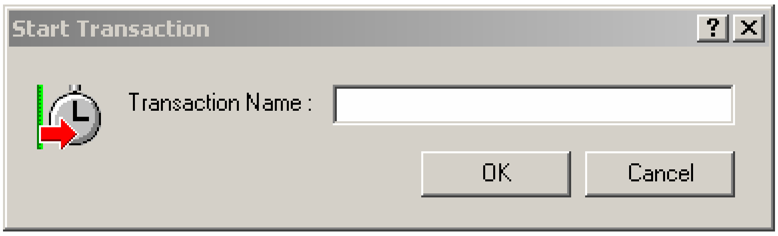
事务（Transaction）：为了衡量服务器的性能，我们需要定义事务。比如：我们在脚本中有一个数据查询操作，为了衡量服务器执行查询操作的性能，我们把这个操作定义为一个事务，这样在运行测试脚本时，LoadRunner 运行到该事务的开始点时，LoadRunner 就会开始计时，直到运行到该事务的结束点，计时结束。这个事务的运行时间在结果中会有反映。

插入事务操作可以在录制过程中进行，也可以在录制结束后进行。LoadRunner 运行在脚本中插入不限数量的事务。

具体的操作方法如下：在需要定义事务的操作前面，通过菜单或者工具栏插入。

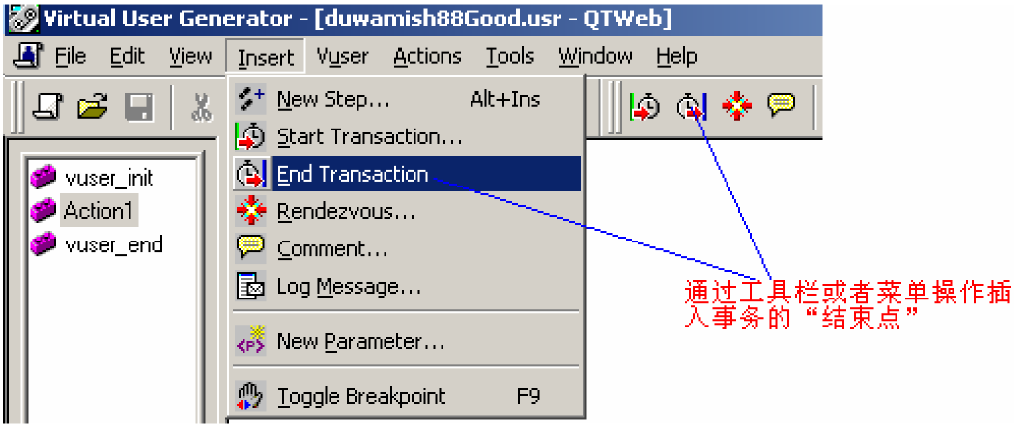


出现以下对话框：

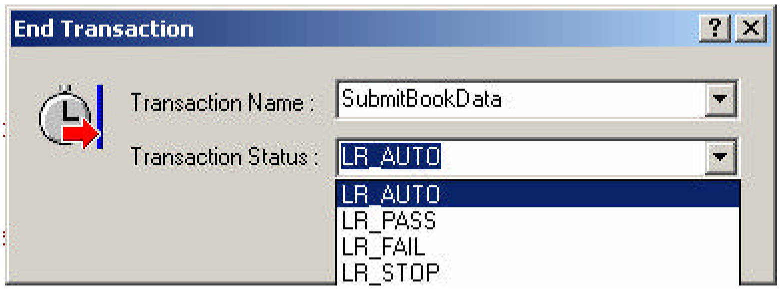


输入该事务的名称。注意：事务的名称最好要有意义，能够清楚的说明该事务完成的动作。

插入事务的开始点后，下面需要在需要定义事务的操作后面插入事务的“结束点”。同样可以通过菜单或者工具栏插入。



出现以下对话框：



默认情况下，事务的名称列出最近的一个事务名称。一般情况下，事务名称不用修改。

事务的状态默认情况下是 LR_AUTO。一般情况下，我们也不需要修改，除非在手工编写代码时，有可能需要手动设置事务的状态。

脚本中事务的代码如下：

```
lr_start_transaction("SubmitBookData");

/*
 * 中间代码是具体事务的操作
 */

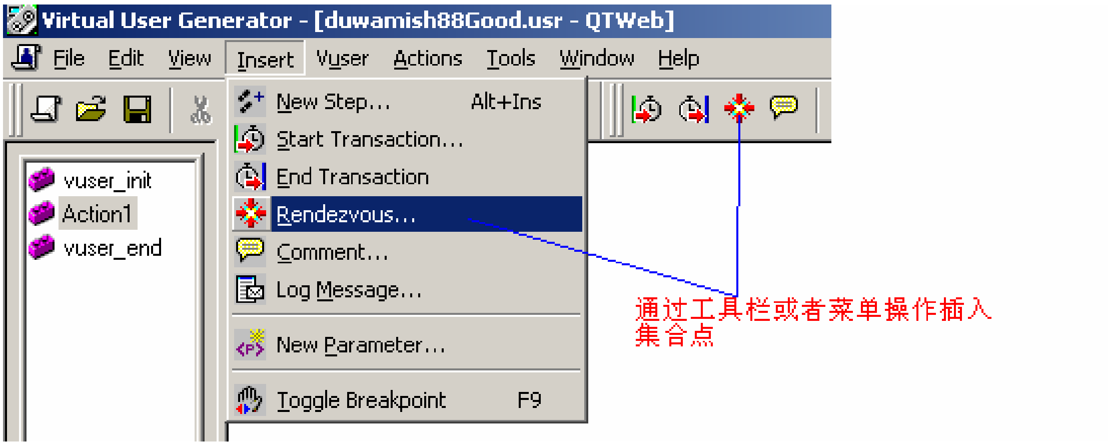
lr_end_transaction("SubmitBookData", LR_AUTO);
```

5.2.2 插入集合点

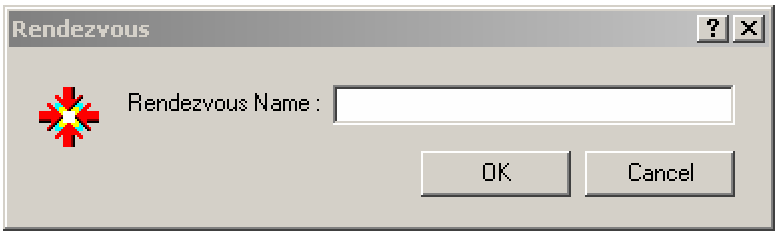
插入集合点是为了衡量在加重负载的情况下服务器的性能情况。在测试计划中，可能会要求系统能够承受 1000 人同时提交数据,在 LoadRunner 中可以通过在提交数据操作前面加入集合点，这样当虚拟用户运行到提交数据的集合点时，LoadRunner 就会检查同时有多少用户运行到集合点,如果不到 1000 人,LoadRunner 就会命令已经到集合点的用户在此等待，当在集合点等待的用户达到 1000 人时，LoadRunner 命令 1000 人同时去提交数据，从而达到测试计划中的需求。

注意：集合点经常和事务结合起来使用。集合点只能插入到 Action 部分，vuser_init 和 vuser_end 中不能插入集合点。

具体的操作方法如下：在需要插入集合点的前面，通过菜单或者工具栏操作



出现对话框



输入该集合点的名称。注意：集合点的名称最好要有意义，能够清楚的说明该集合点完成的动作。

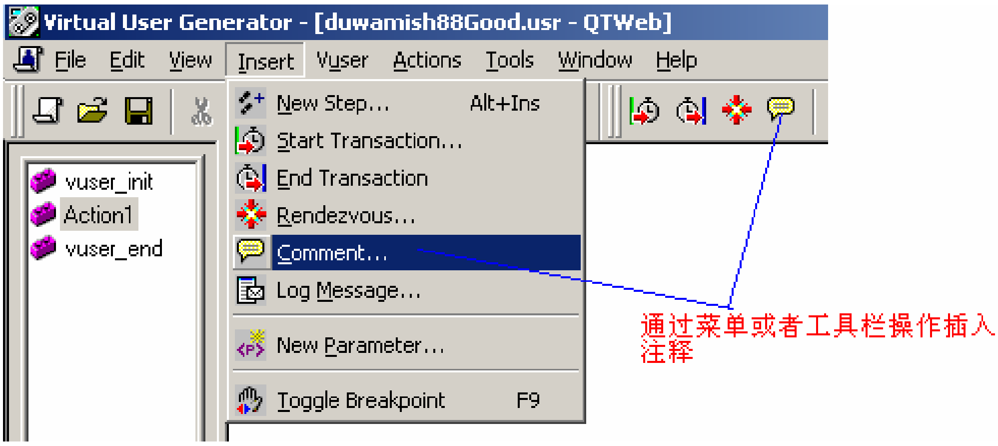
脚本中集合点的代码如下：

```
lr_rendezvous("SubmitQueryData");
```

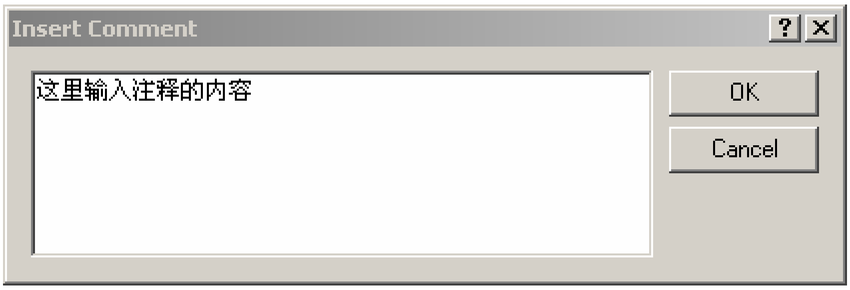
5.2.3 插入注释

注释的作用就不多说了，不过插入注释最好是在录制过程中。

具体的操作方法如下：在需要插入注释的前面，通过菜单或者工具栏操作



出现对话框



脚本中注释的代码如下：

```
/*
 * 这里输入注释的内容
 */
```

5.2.4 参数化输入

如果用户在录制脚本过程中，填写提交了一些数据，比如要增加数据库记录。这些操作都被记录到了脚本中。当多个虚拟用户运行脚本时，都会提交相同的记录，这样不符合实际的运行情况，而且有可能引起冲突。为了更加真实的模拟实际环境，需要各种各样的输入。参数化输入是一种不错的方法。

用参数表示用户的脚本有两个优点：

① 可以使脚本的长度变短。

② 可以使用不同的数值来测试你的脚本。例如，如果你企图搜索不同名称的图书，你仅仅需要写提交函数一次。在回放的过程中，你可以使用不同的参数值，而不只搜索一个特定名称的值。

参数化包含以下两项任务：

① 在脚本中用参数取代常量值。

② 设置参数的属性以及数据源。

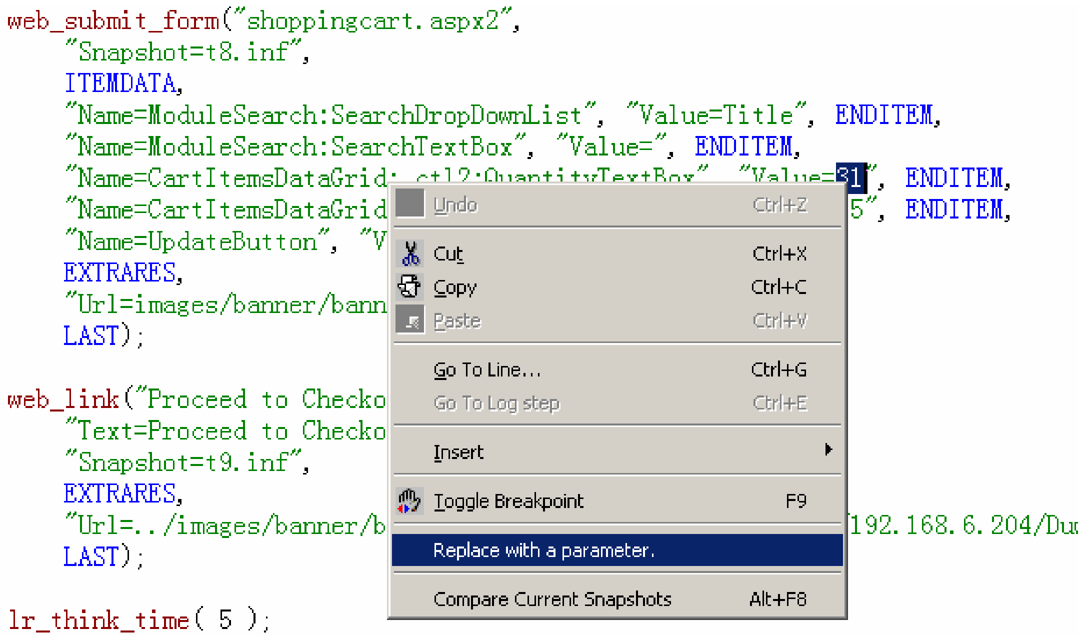
参数化仅可以用于一个函数中的参量。你不能用参数表示非函数参数的字符串。

另外，不是所有的函数都可以参数化的。

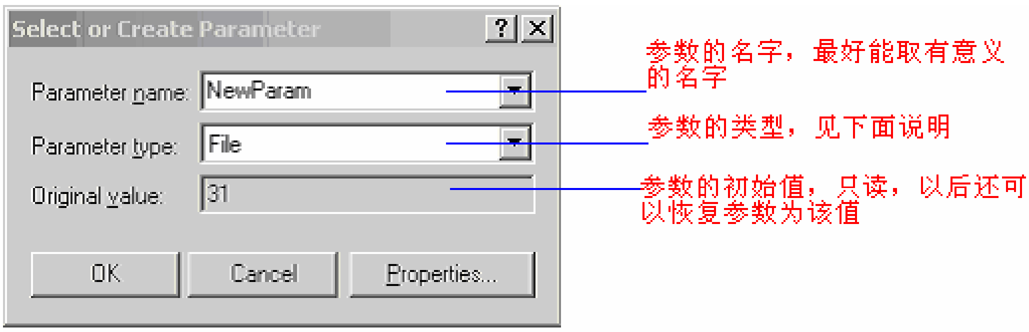
参数化输入的讲解，我们采用一个例子的方式来进行。

```
web_submit_form("shoppingcart.aspx2",
    "Snapshot=t8.inf",
    ITEMDATA,
    "Name=ModuleSearch:SearchDropDownList", "Value=Title", ENDITEM,
    "Name=ModuleSearch:SearchTextBox", "Value=", ENDITEM,
    "Name=CartItemsDataGrid:_ctl2:QuantityTextBox", "Value=31", ENDITEM,
    "Name=CartItemsDataGrid:_ctl3:QuantityTextBox", "Value=15", ENDITEM,
    "Name=UpdateButton", "Value=Update", ENDITEM,
    EXTRARES,
    "Url=images/banner/bannerslice.gif", ENDITEM,
    LAST);
```

假如有以上的一个提交数据的窗体，我们想参数化高亮显示的部分（31**）。操作方法很简单，我们只要选中“** 31 **”，然后点鼠标右键**



选择“ Replace with a parameter. **”，出现以下窗口：**

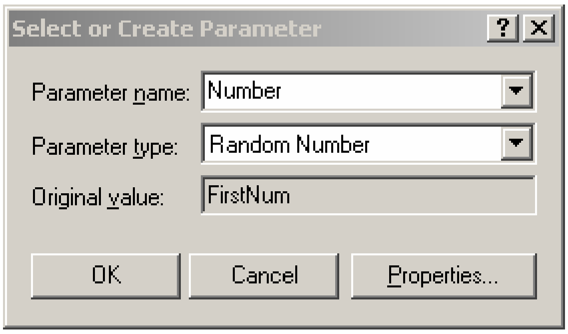


下面我们重点介绍一下参数的类型。

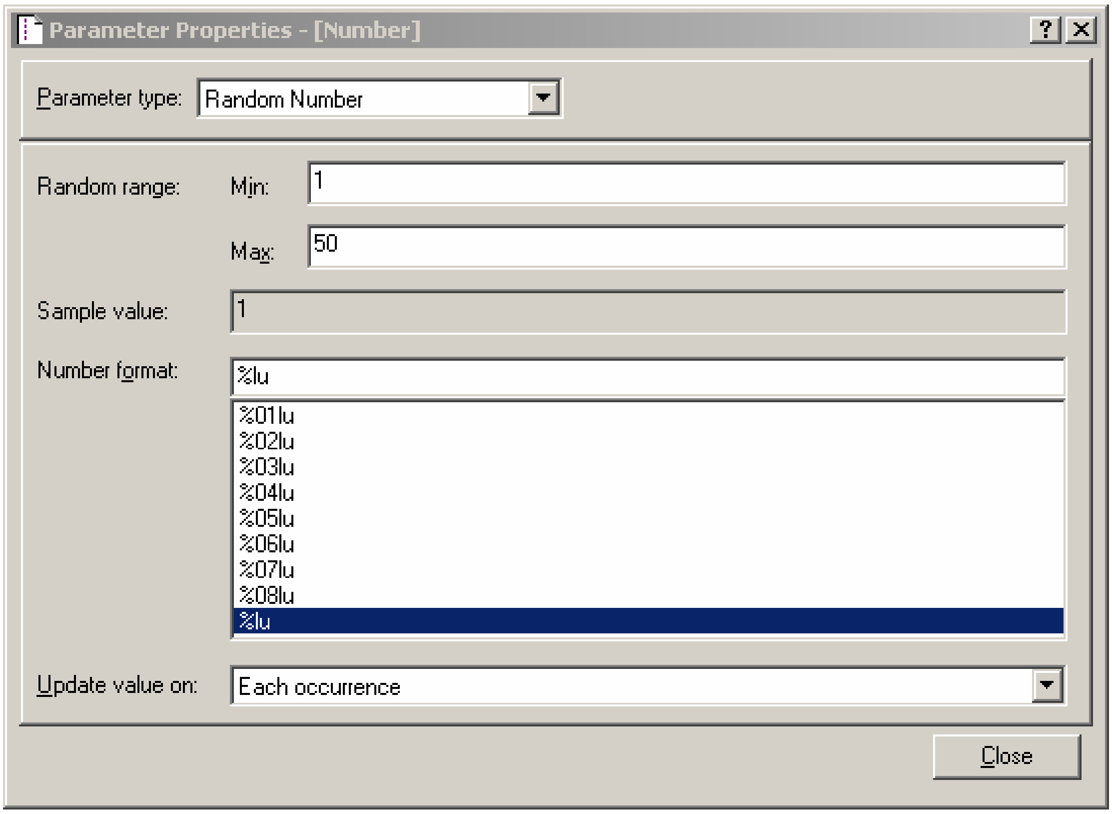
- DateTime**：很简单，在需要输入日期/时间的地方，可以用** DateTime **类型来替代。其属性设置也很简单，选择一种格式即可。当然也可以定制格式。**
- Group Name **:暂时不知道何处能用到，但设置比较简单。在实际运行中，**LoadRunner **使用该虚拟用户所在的** Vuser Group **来代替。但是在** VuGen **中运行时，**Group Name **将会是** None

- Load Generator Name：在实际运行中，LoadRunner 使用该虚拟用户所在 Load Generator 的机器名来代替。
- Iteration Number：在实际运行中，LoadRunner 使用该测试脚本当前循环的次数来代替。
- Random Number：随机数。很简单。在属性设置中可以设置产生随机数的范围
- Unique Number：唯一的数。在属性设置中可以设置第一个数以及递增的数的大小。**注意：使用该参数类型必须注意可以接受的最大数。例如：某个文本框能接受的最大数为 99。当使用该参数类型时，设置第一个数为 1，递增的数为 1，但 100 个虚拟用户同时运行时，第 100 个虚拟用户输入的将是 100，这样脚本运行将会出错。注意：这里说的递增意思是各个用户取第一个值的递增数，每个用户相邻的两次循环之间的差值为 1。举例说明：假如起始数为 1，递增为 5，那么第一个用户第一次循环取值 1，第二次循环取值 2；第二个用户第一次循环取值为 6，第二次为 7；依次类推。**
- Vuser ID：设置比较简单。在实际运行中，LoadRunner 使用该虚拟用户的 ID 来代替，该 ID 是由 Controller 来控制的。但是在 VuGen 中运行时，Vuser ID 将会是 –1。
- File：需要在属性设置中编辑文件，添加内容，也可以从现成的数据库中取数据（下面我们将会介绍）
- User Defined Function：从用户开发的 dll 文件提取数据。就目前我认为，这种方式没有必要。VuGen 支持 C 语言的语法，在 VuGen 中重新编写类似的函数应该不难。（**一家之言，仅供参考**）

上面的例子中，我们取随机数即可。



点"Properties….."按钮，进行属性设置窗口



添入随机数的取值范围为（1-50），选择一种数据格式。在 Update Value on 中有以下几个选项：

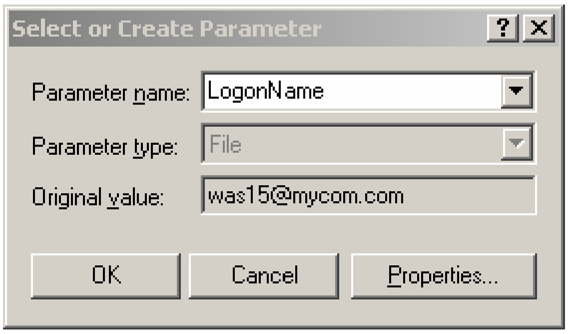
- Each Occurrence：在运行时，每遇到一次该参数，便会取一个新的值
- Each iteration：运行时，在每一次循环中都取相同的值
- Once：运行时，在每次循环中，该参数只取一次值

这里我们用的是随机数，选择 Each Occurrence 非常合适。

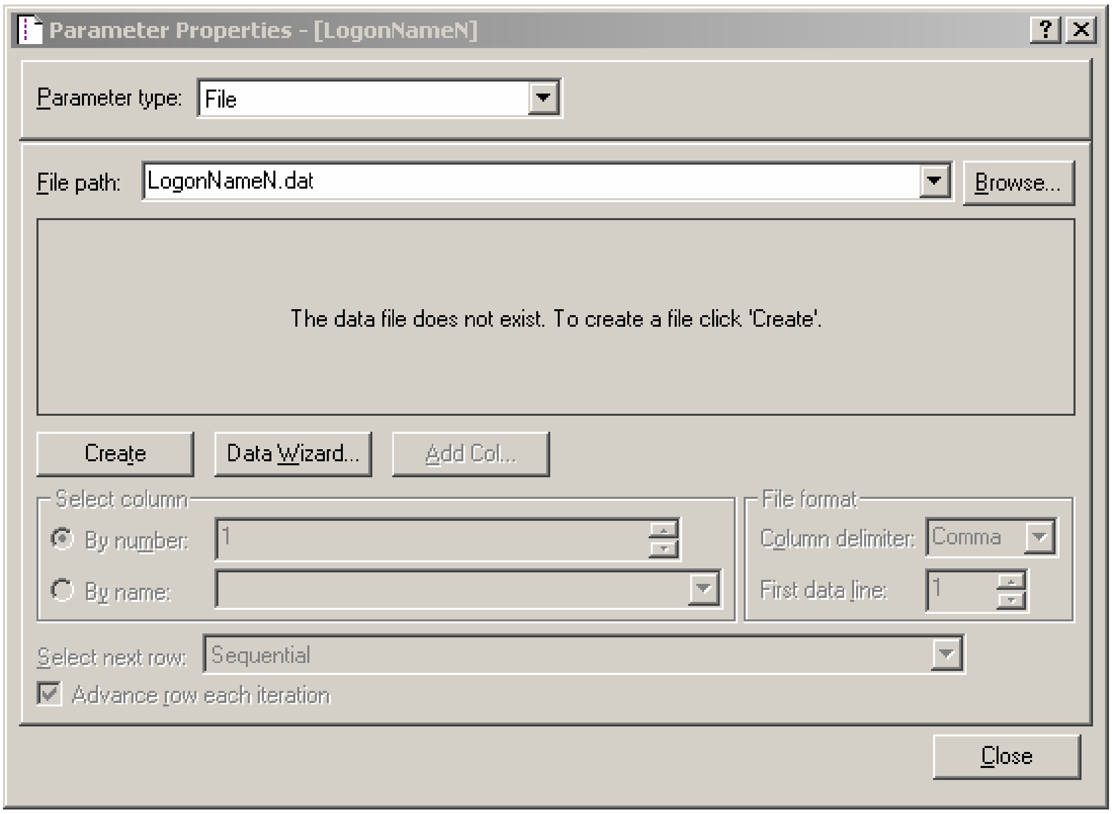
下面我们再举一个 file 的例子。我们要用数据库中的用户名来参数化登陆用户名。

```
web_submit_form("logon.aspx",
    "Snapshot=t10.inf",
    ITEMDATA,
    "Name=LogonEmailTextBox", "Value=was15@mycom.com", ENDITEM,
    "Name=LogonPasswordTextBox", "Value=was", ENDITEM,
    "Name=LogonButton", "Value=Logon", ENDITEM,
    EXTRARES,
    "Url=../images/banner/bannerslice.gif", "Referer=http://192.168.6.204/Duwa
    LAST);
```

选择要参数化的数据，右键，选择“Replace with a parameter.”，出现以下窗口：

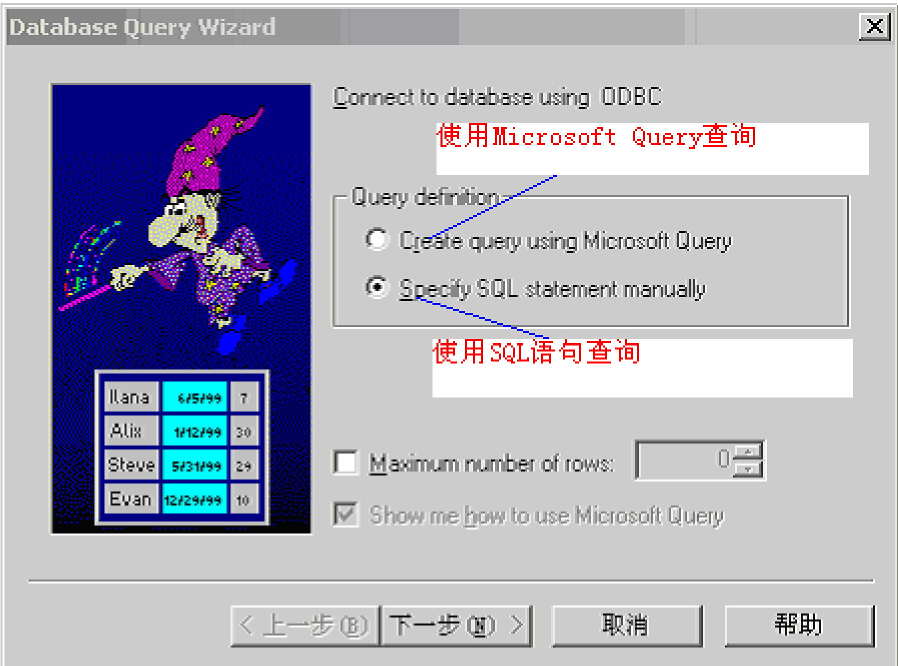


点“ Properties….. ”，按钮，出现以下窗口

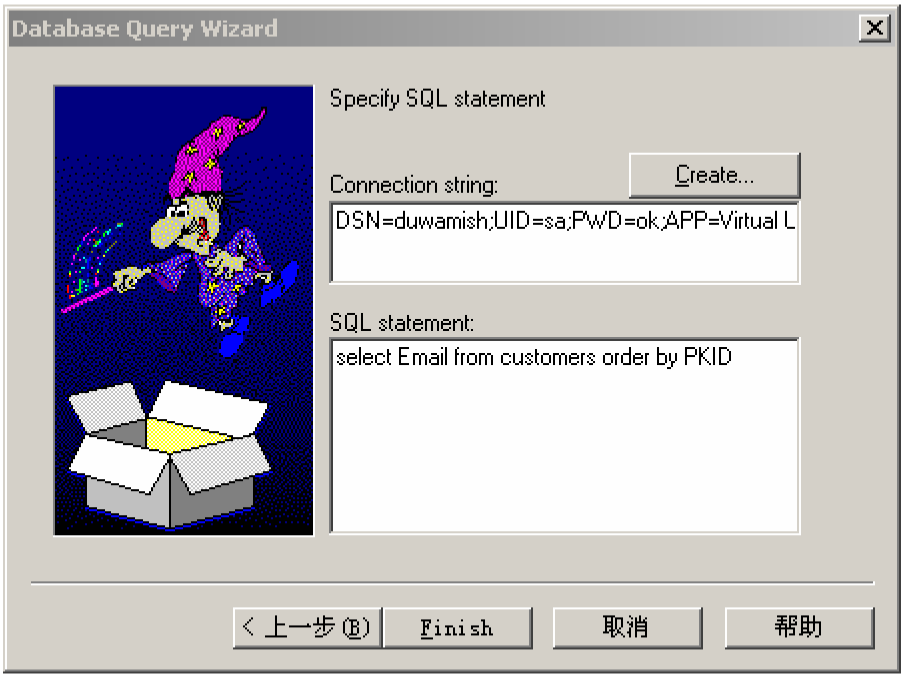


注意：参数的文件名不要使用 con.dat、pm.dat 或者 lpt*.dat 等系统装置名

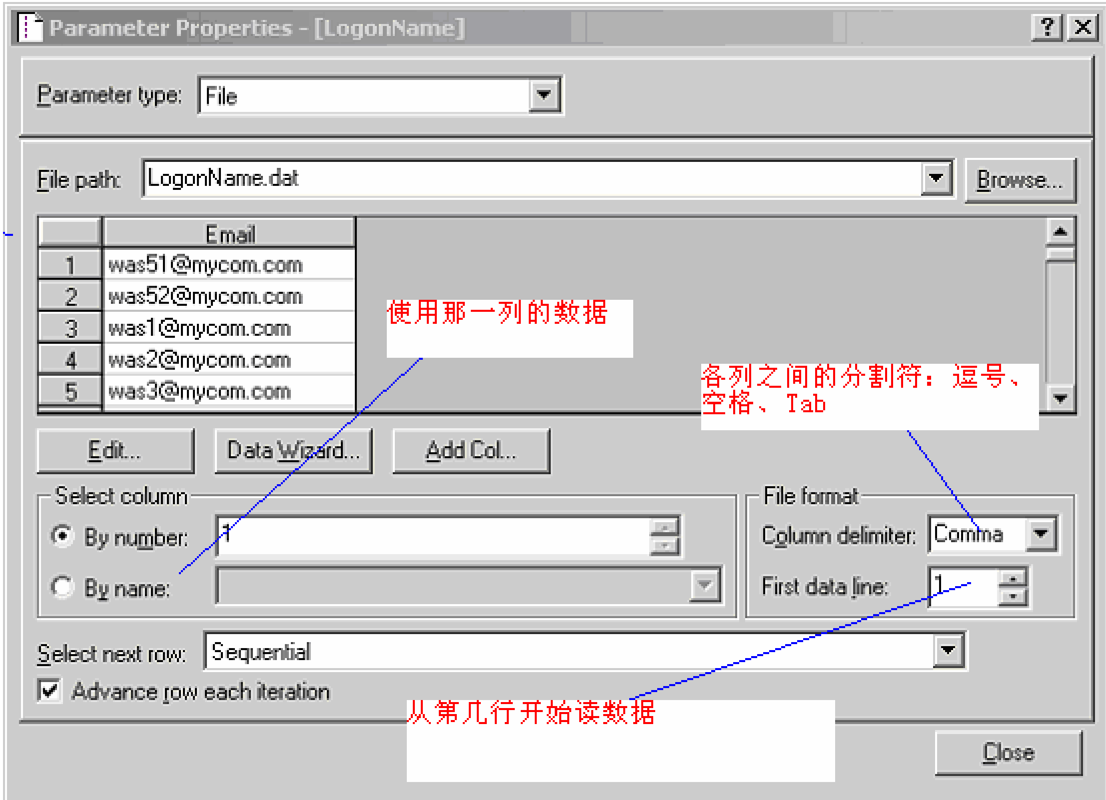
下面我们将会连接数据库，从数据表中选择用户名。点“ Data Wizard ”按钮



使用第 2 项，下一步



添入连接字符串和 SQL 语句后，点 Finish 按钮，出现查询结果。



提醒：在参数数据显示区，最多只能看到 100 行，如果数据超过 100 行，只能点“Edit”按钮，进入记事本看。

“Select next row　”有以下几种选择：

- Sequential：按照顺序一行行的读取。每一个虚拟用户都会按照相同的顺序读取
- Random：在每次循环里随机的读取一个，但是在循环中一直保持不变

- Unique：唯一的数。注意：使用该类型必须注意数据表有足够多的数。比如 Controller 中设定 20 个虚拟用户进行 5 次循环，那么编号为 1 的虚拟用户取前 5 个数，编号为 2 的虚拟用户取 6-10 的数，依次类推，这样数据表中至少要有 100 个数据，否则 Controller 运行过程中会返回一个错误。
- Same Line As 某个参数（比如 Name）：和前面定义的参数 Name 取同行的记录。通常用在有关联性的数据上面。

我们这里取值 Sequential 即可。

Advance row each iteration 选中即可，表示每一次循环都往前走一行。

手工输入数据比较简单，这里就不再单独介绍了。

5.2.5 插入函数

VuGen 中可以使用 C 语言中比较标准的函数和数据类型，语法和 C 语言相同。下面简单介绍一下比较常用的函数和数据类型。

1. 控制脚本流程

   if { } else { }

   for{ }

   while{ }

   ……………

   总之 C 语言的控制流程的语句这里都可以直接使用
2. 字符串函数

   由于在 VuGen 脚本中使用最多的还是字符串，所以字符串函数在脚本中使用非常频繁。具体的语法请参考帮助说明。

   strcmp　　比较两个字符串

   strcat　　连接两个字符串

   strcpy　　拷贝字符串

   ……………..

   注意：在 VuGen 中，以 char*声明的字符串是只读的，如果试图给 char*类型的字符串赋值的话，编译会通过，但在运行时会产生"Access Violation"的错误。解决这类问题，就是把字符串声明为字符数组，比如 char[100]。
3. 输出函数

   输出函数在调试脚本时非常有用。

   lr_output_message　　输出一条消息

   ………………..
4. LoadRunner 提供的标准函数

   lr_eval_string　　该函数功能是得到参数（参数化输入中）当前的值

   　exg:　**lr_output_message("temp = %s", lr_eval_string("{WCSParam2}"));**

   lr_save_string　　该函数功能是把一个字符串保存到参数中

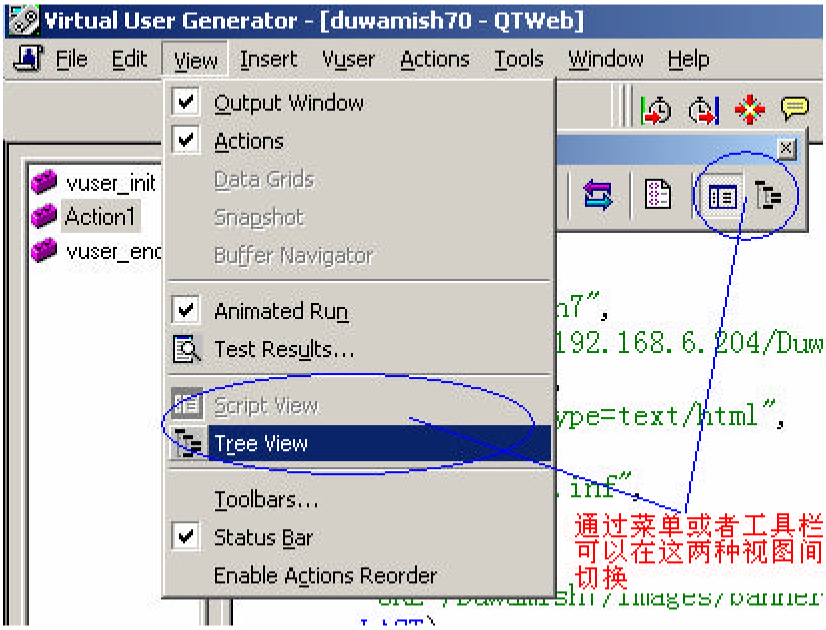
   　exg:　**lr_save_string("439","WCSParam3");**

5.2.6 插入 Text/Imag 检查点

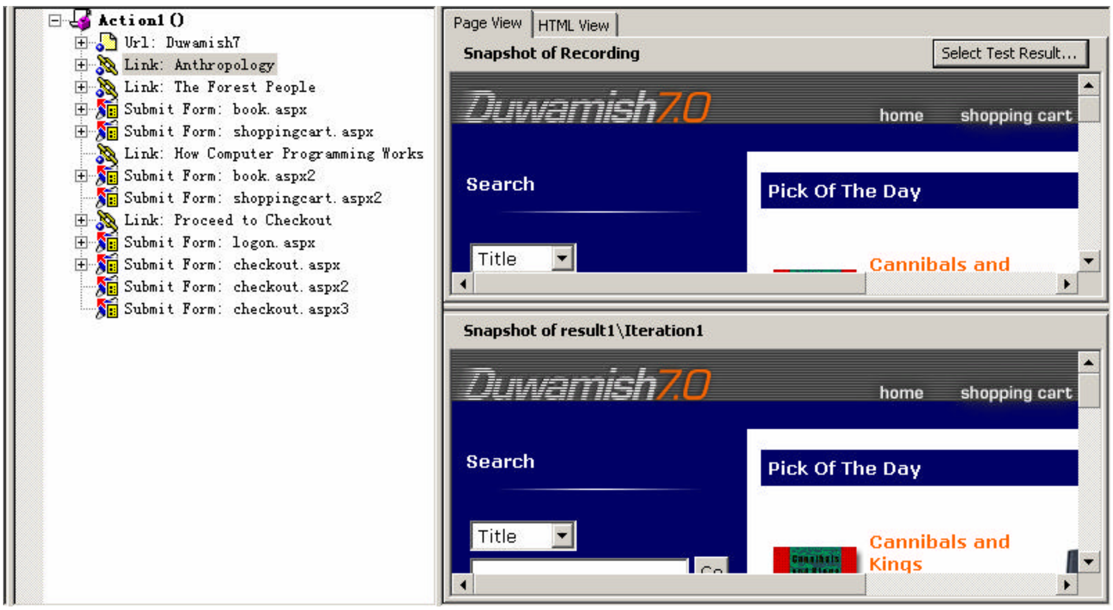
在进行压力测试时，为了检查 Web 服务器返回的网页是否正确，VuGen 允许我们插入 Text/Imag 检查点，这些检查点验证网页上是否存在指定的 Text 或者 Imag，还可以测试在比较大的压力测试环境中，被测的网站功能是否保持正确。检查点的含义和 WinRunner 中的检查点功能基本上一致，这里就不再作过多的说明。

VuGen 在测试 Web 时，有两种视图方式：TreeView/Script View。前面我们见到的一直都是 Script View。在插入 Text/Imag 检查点时，我觉得用 TreeView 视图会比较方便。

在这种视图之间切换，可以通过菜单或者工具栏的方式

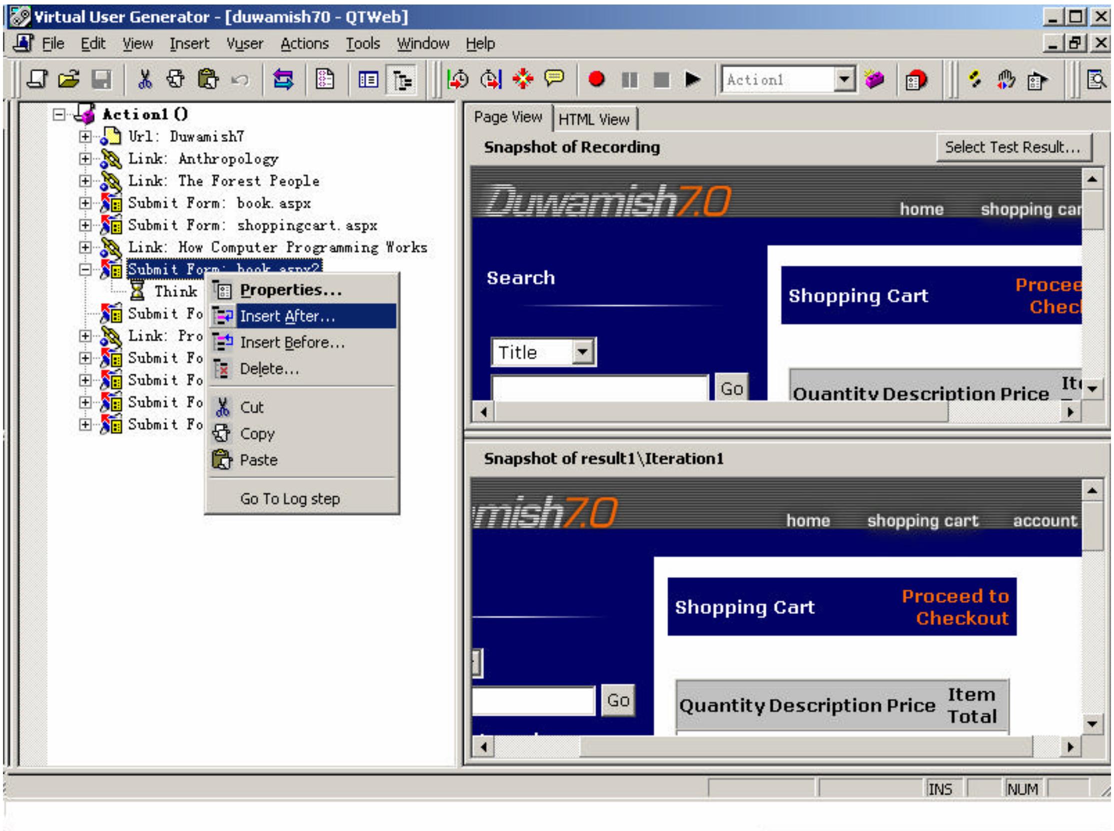


现在我们就切换到 TreeView 视图

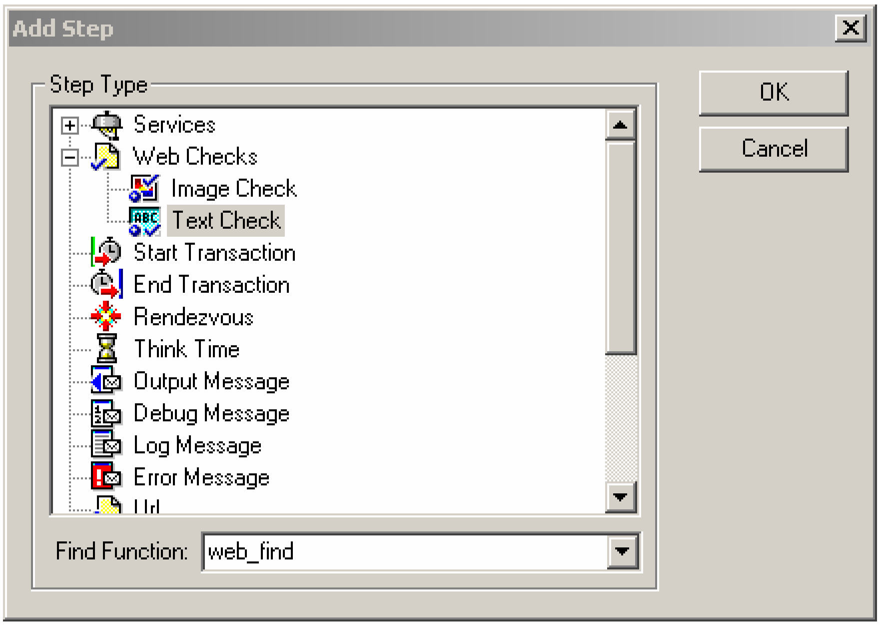


插入检查点的步骤比较简单。添加 Text/Imag 检查点，可以在录制过程中，也可以在录制完

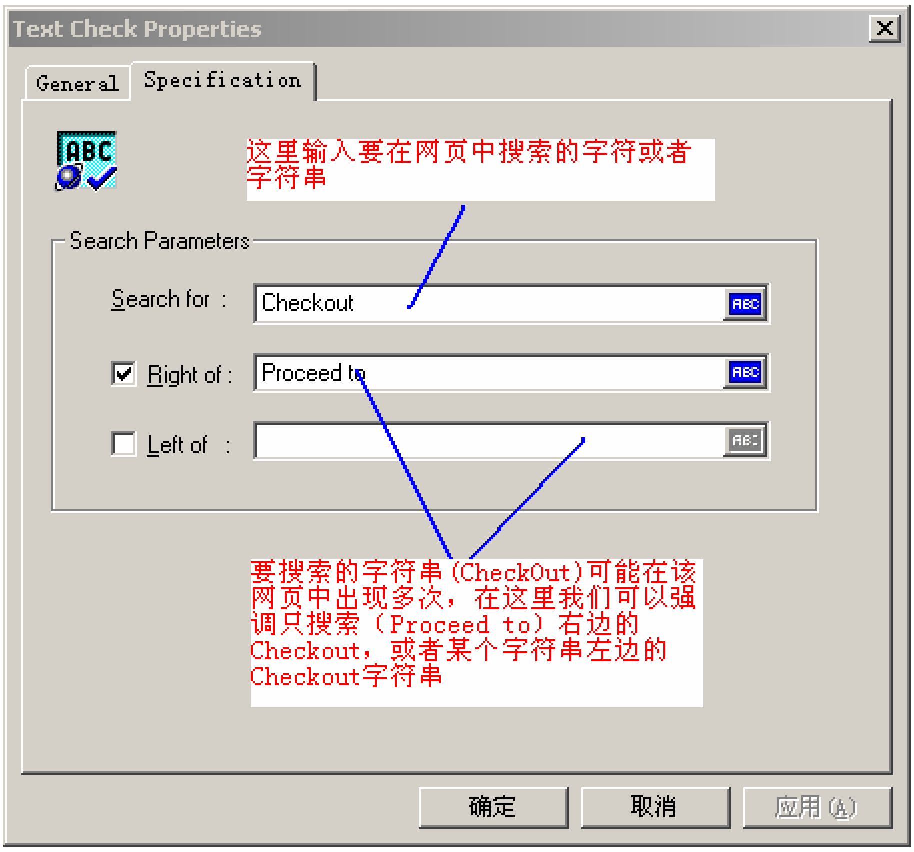
成后进行。**推荐最好能在录制过程中添加 Text/Imag 检查点。**
先在树形菜单中选择需要插入检查点的一项，然后点鼠标右键，选择将检查点插到该操作执行前还是该操作执行后。如果在该操作执行前，则选择“Insert Before”，否则选择“Insert After”。



然后弹出对话框，如下，选择“Text Check”(这里以 Text 检查点为例说明)

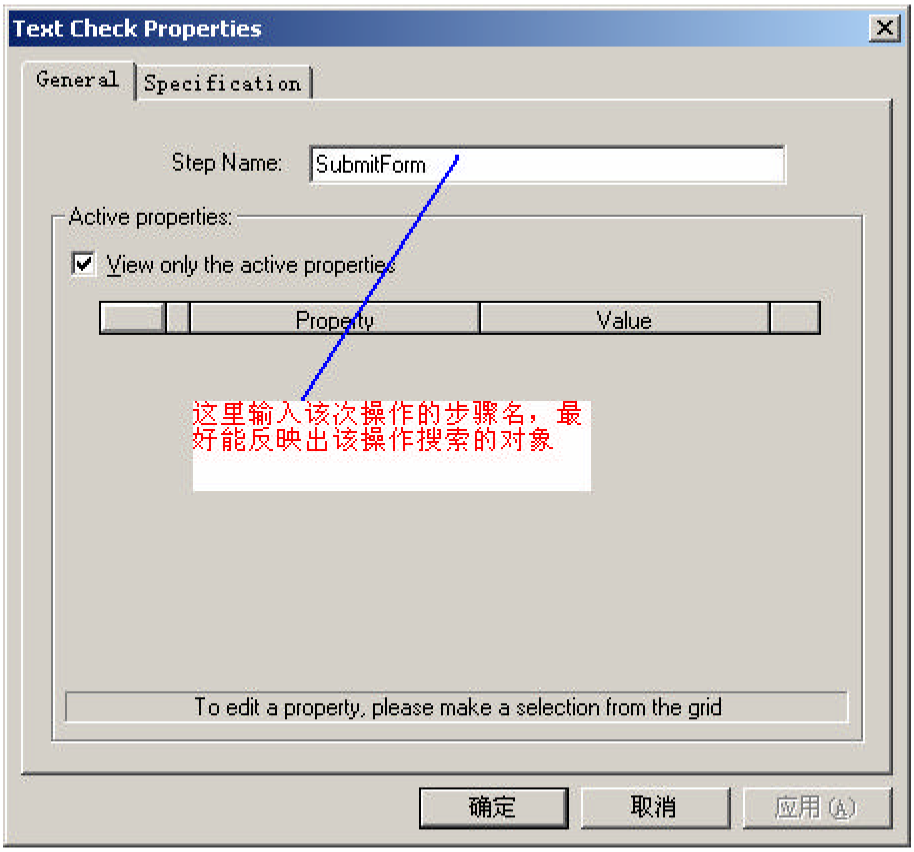


OK 后，出现 Text Check Properties 对话框



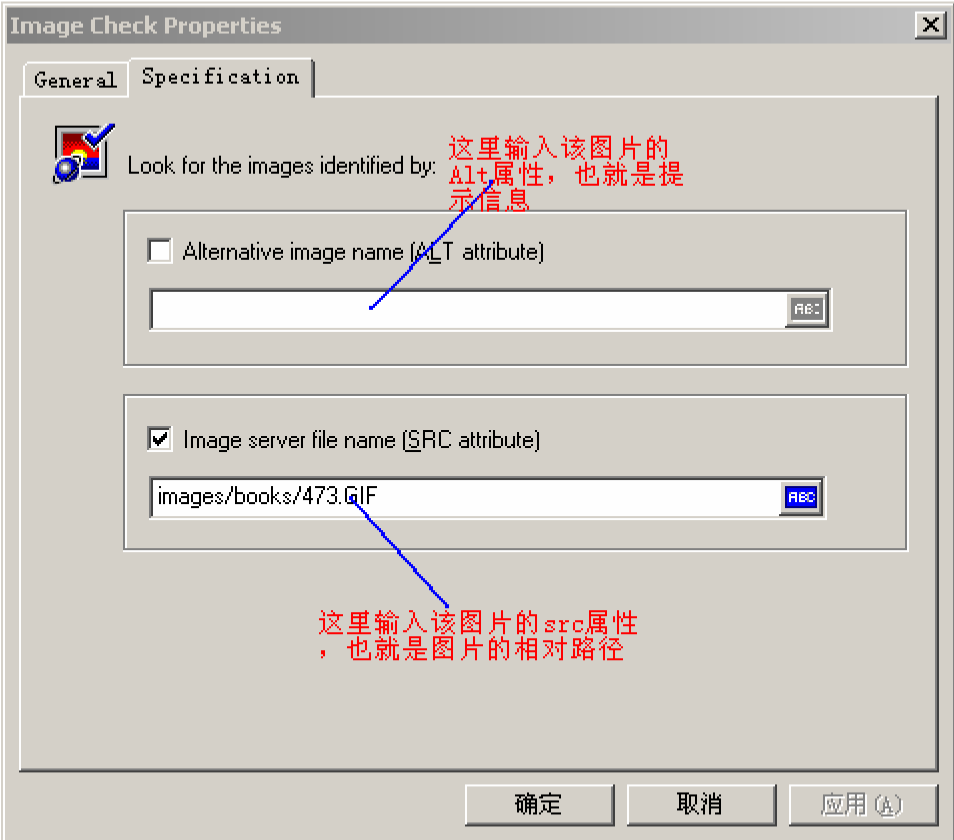
注意：这里要搜索的字符串可以使用正则表达式。

然后切换到"General"标签页



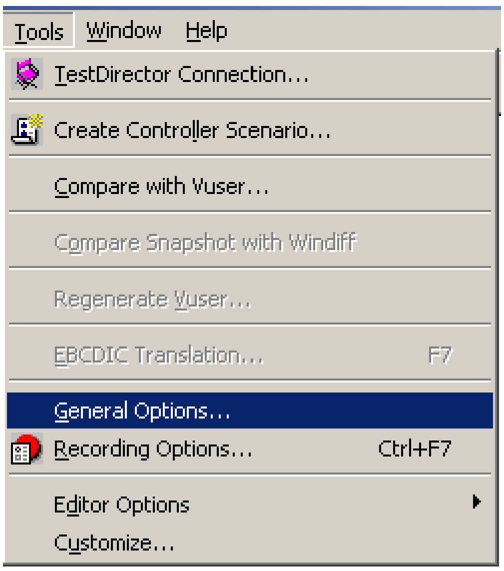
确定后，即可完成添加 Text 检查点的任务。

添加 Imag 检查点的操作步骤和 Text 检查点差不多，这里仅仅对 Imag Check Properties 窗口进行说明。其他的和 Text 检查点类似，不再详细说明。

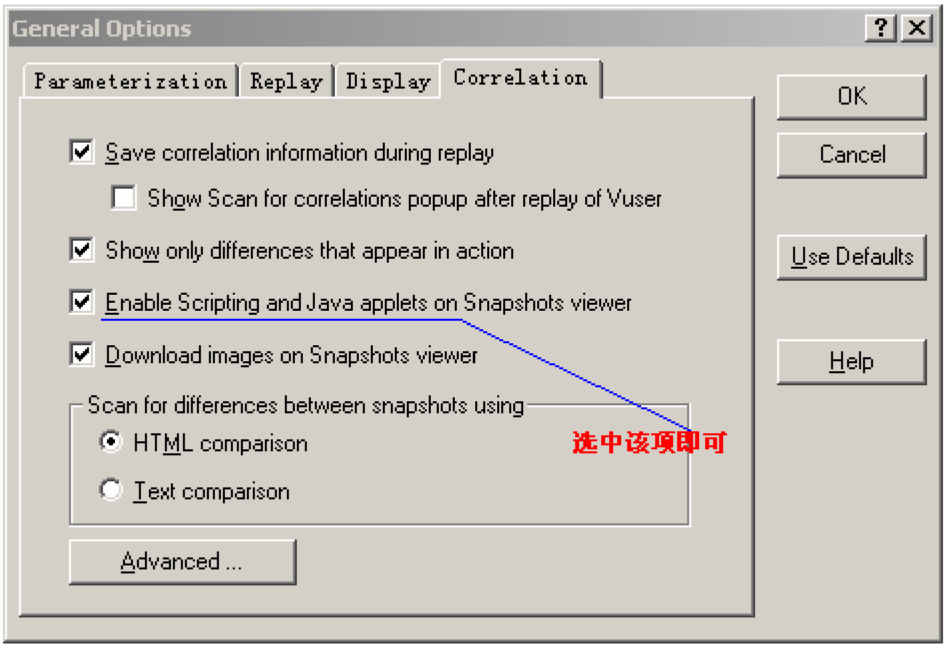


当然 VuGen 还允许插入其他类型的检查点函数，比如 web_reg_find、Web_global_verification 等。而且这里也可以对搜索 Text/Imag 值的参数化，这里就不再一一说明。

注：如果 Web 窗体中包含有 JavaScript 脚本，那么在 TreeView 视图中显示可能会有问题。解决这个问题，可以设置一下。

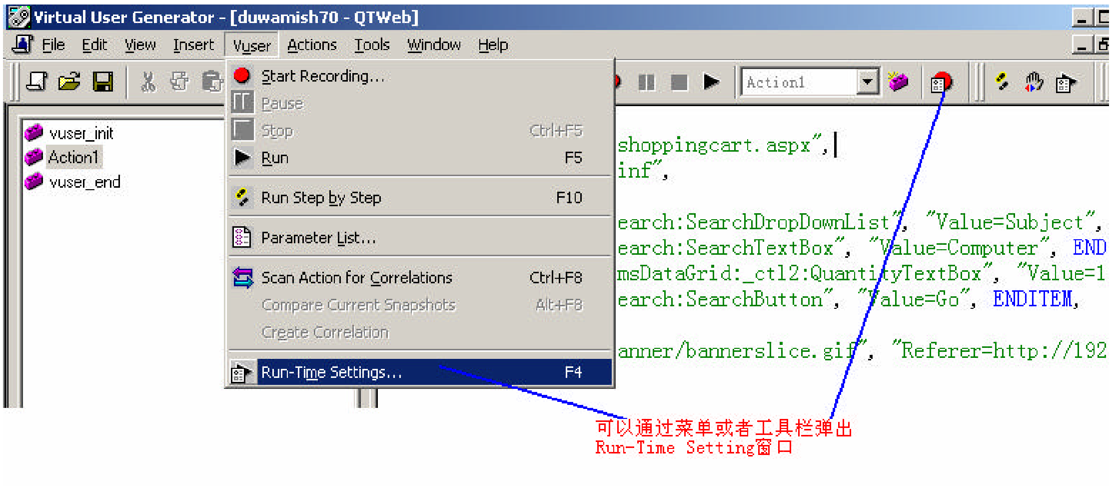


进入设置窗口

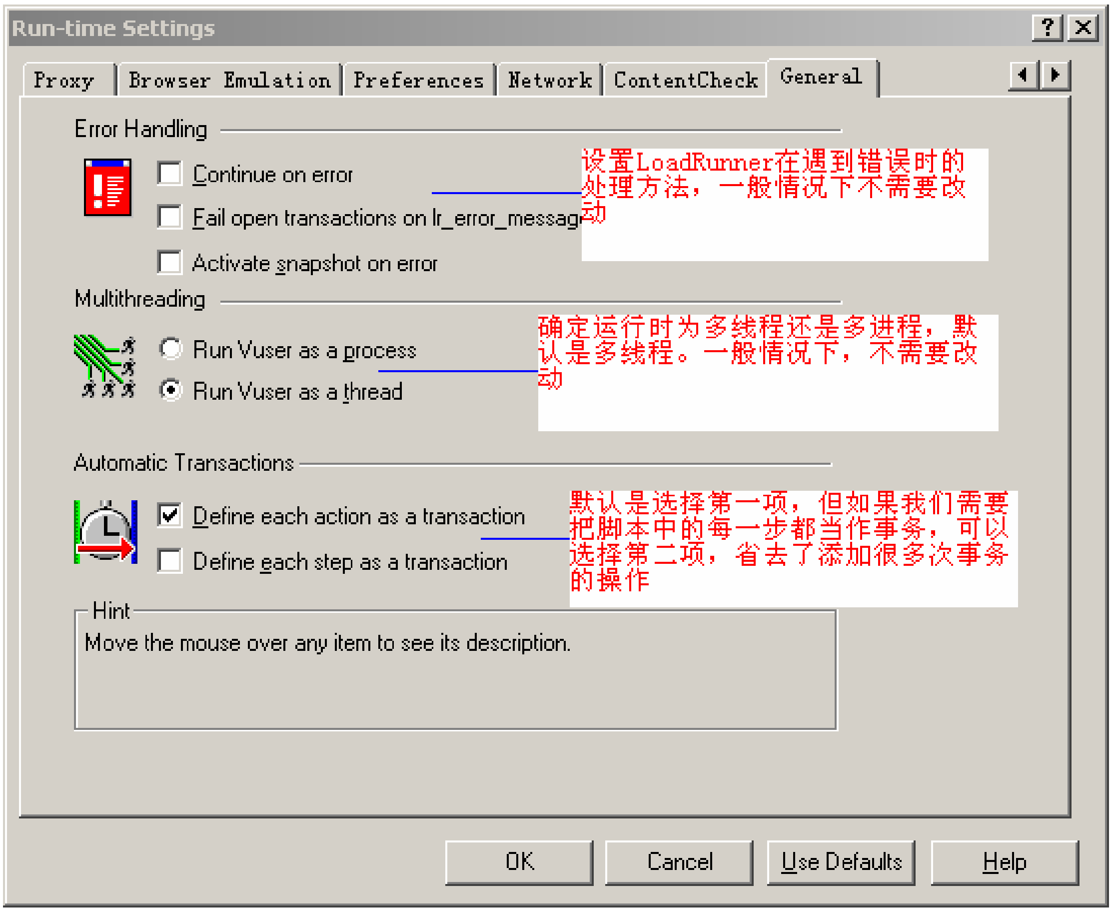


5.3 Run-Time Setting

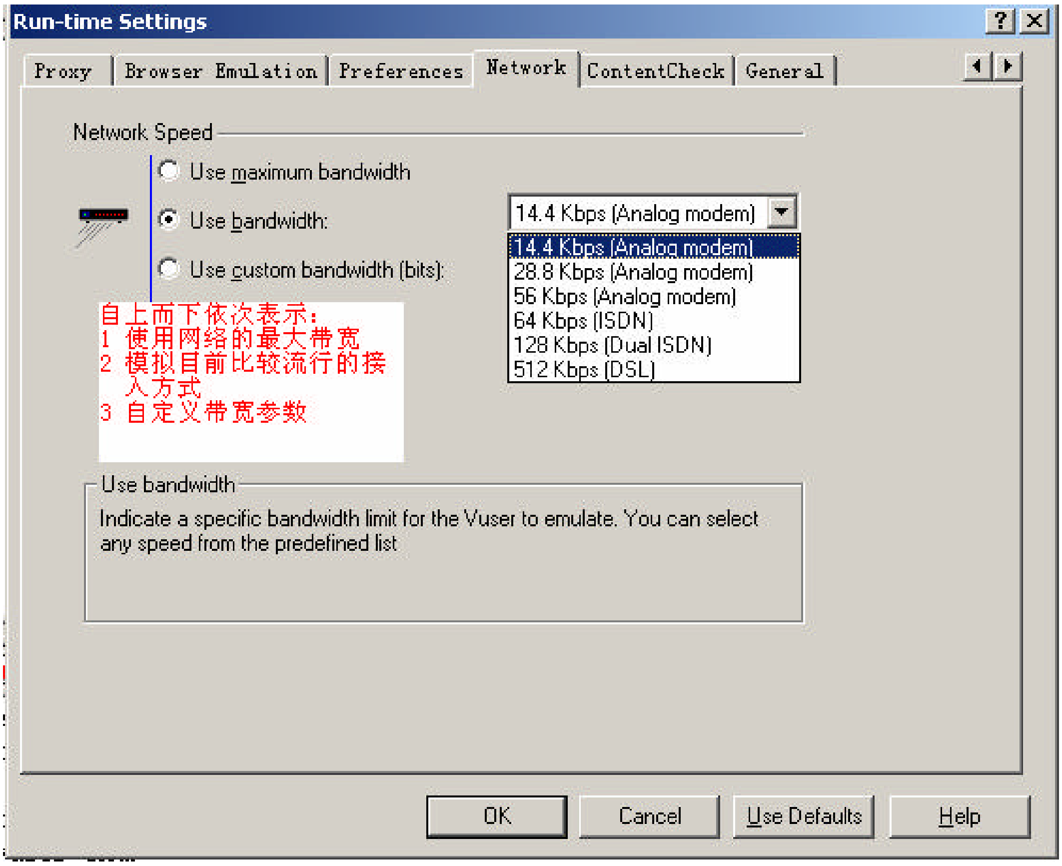
当完善了测试脚本后，需要对 VuGen 的 Run-Time Setting 进行配置。
下面对经常需要设置的几个标签页进行说明。
首先打开 Run-Time Setting 窗口，可以通过菜单或者工具栏进行。



操作后出现 Run-Time Setting 窗口，打开"General"标签页

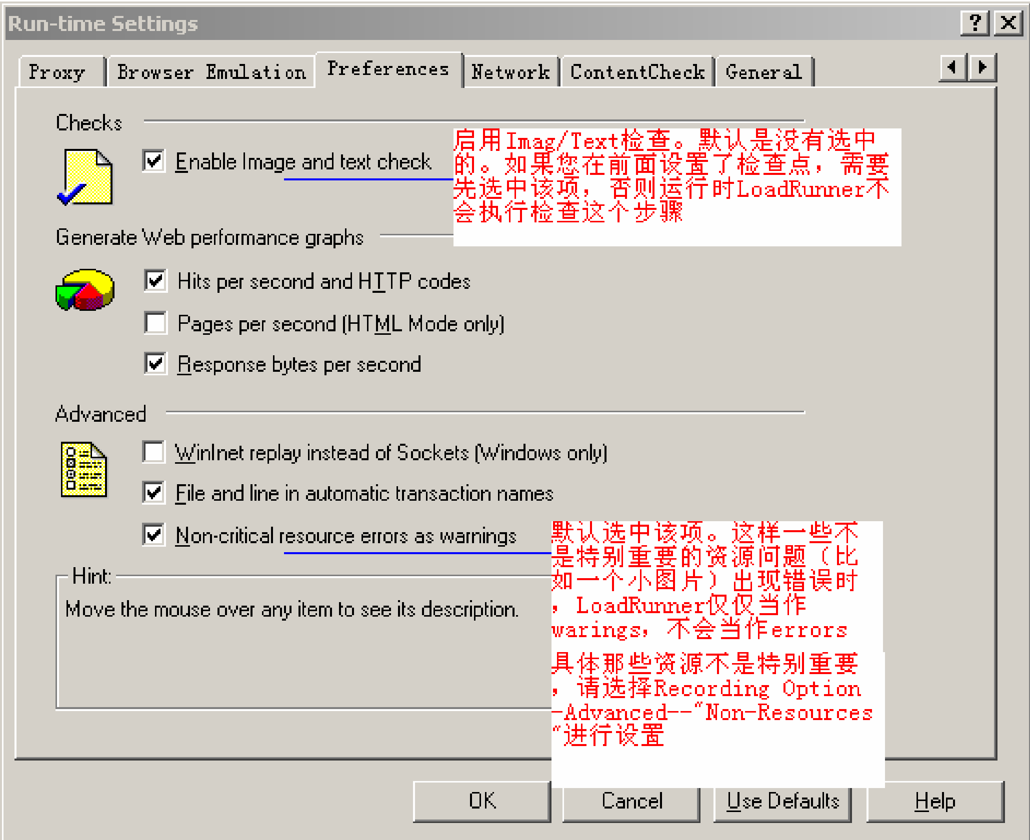


切换到“NetWork”标签页

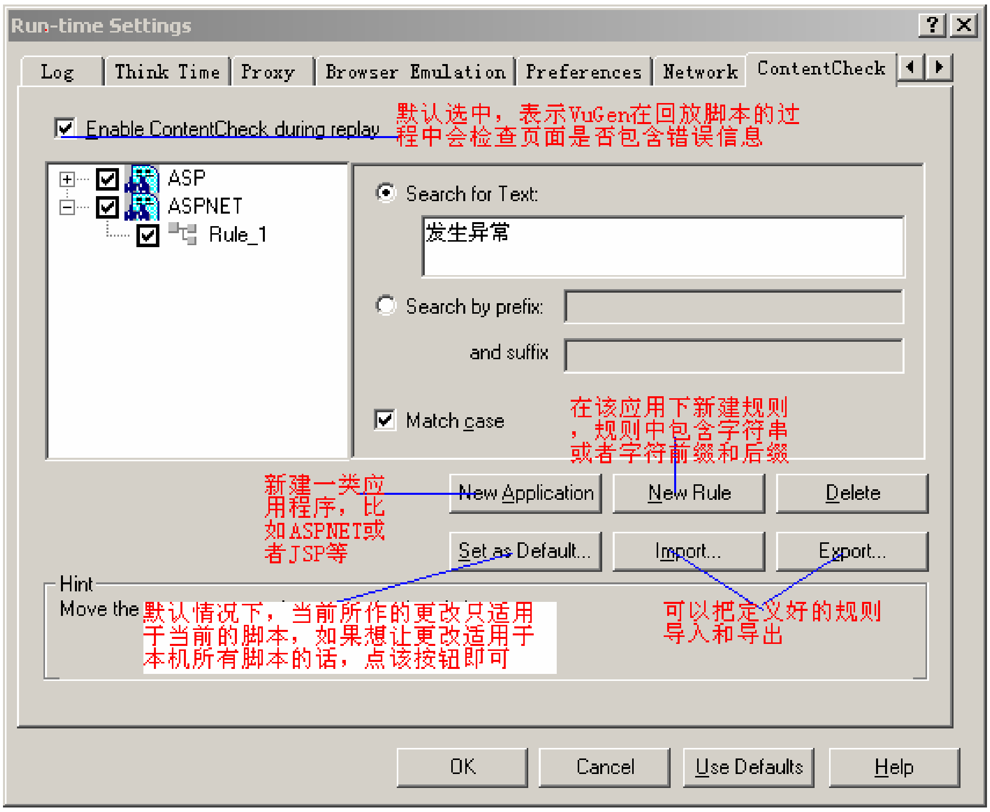


注意：带宽越大，给 Web 服务器造成的压力就越大。

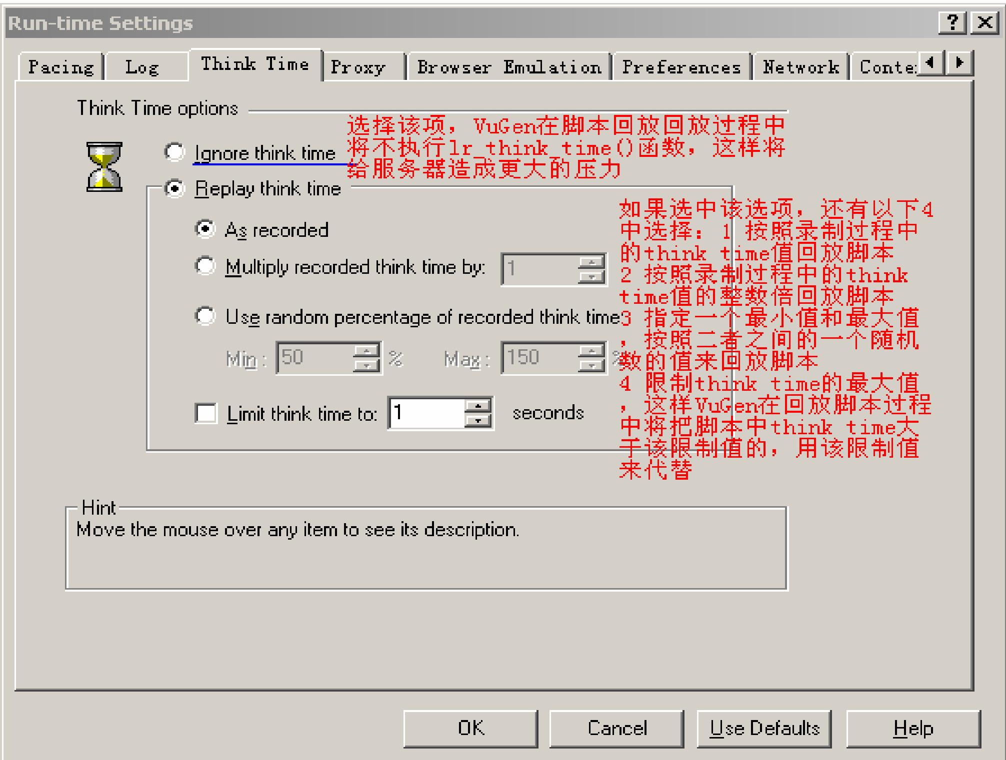
切换到“ Preferences ”标签页，这里仅仅对两个经常需要改动的选项进行说明。



切换到“ ContentCheck ”标签。这里的设置是为了让 VuGen 检测何种页面为错误页面。如果被测的 Web 应用没有使用自定义的错误页面，那么这里不用作更改；如果被测的 Web 应用使用了自定义的错误页面，那么这里需要定义，以便让 VuGen 在运行过程中检测，服务器返回的页面是否包含预定义的字符串，进而判断该页面是否为错误页面。如果是，VuGen 就停止运行，指示运行失败。



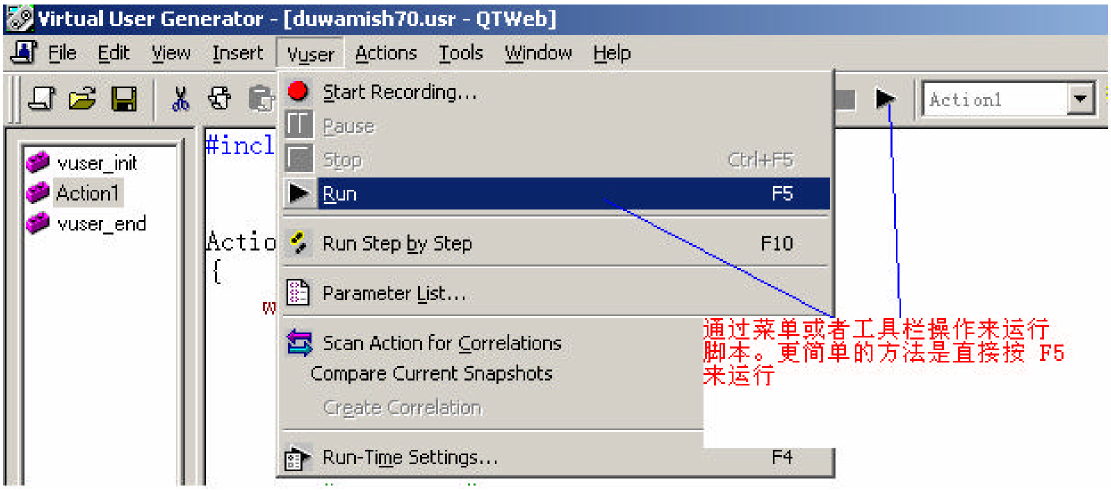
切换到“ Think Time ”标签



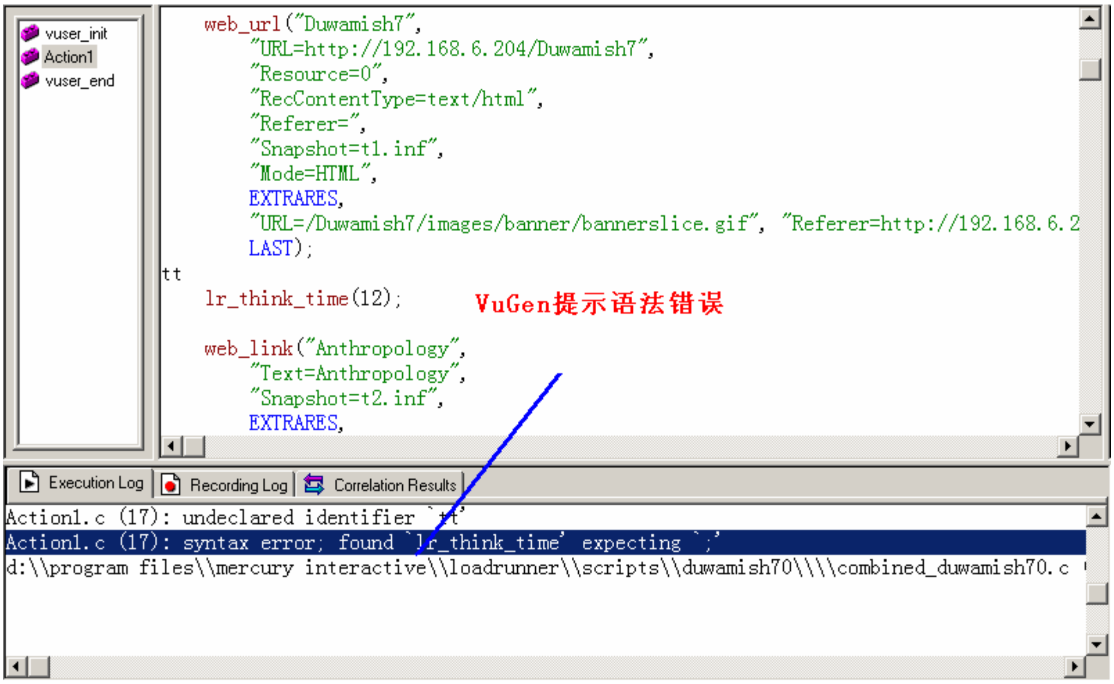
其它的标签设置采用默认值即可，这里不再详细的介绍。

5.4 单机运行测试脚本

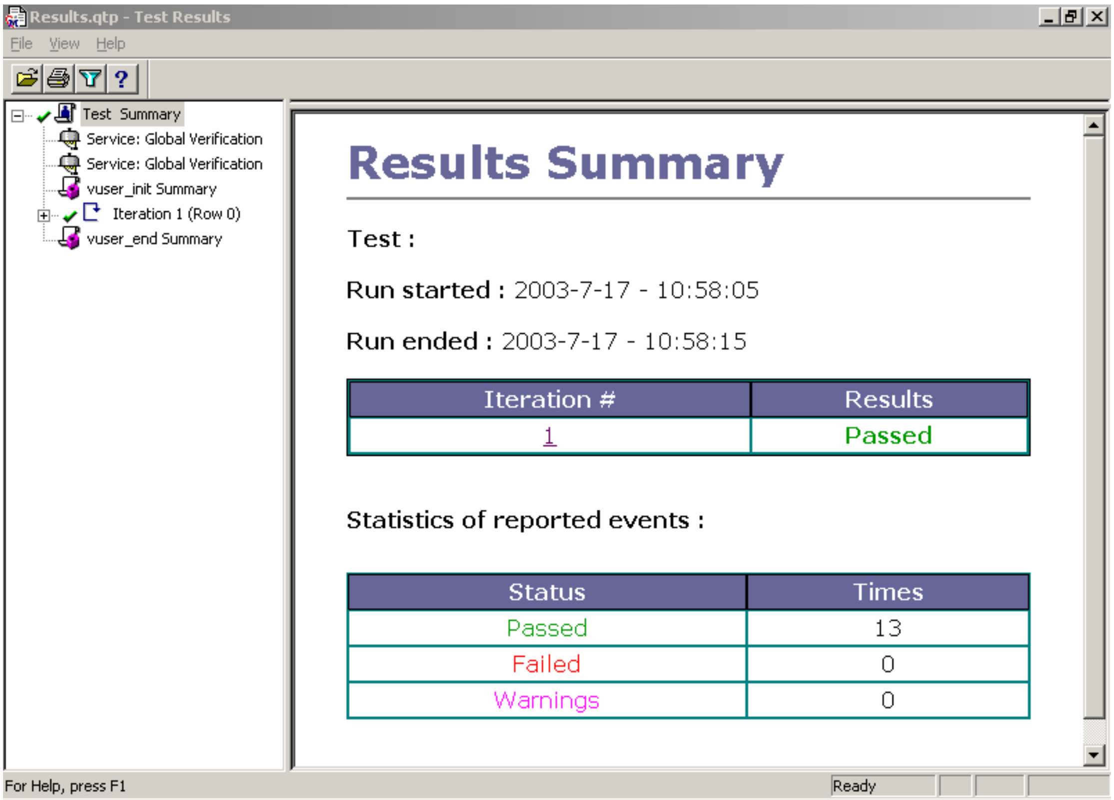
经过以上的各个步骤后，脚本就可以运行了。运行脚本可以通过菜单或者工具栏来操作。



执行“运行”命令后，VuGen 先编译脚本，检查是否有语法等错误。如果有错误，VuGen 将会提示错误。双击错误提示，VuGen 能够定位到出现错误的那一行。为了验证脚本的正确性，我们还可以调试脚本，比如在脚本中加断点等，操作和在 VC 中完全一样，相信大家谁都不会感到陌生。



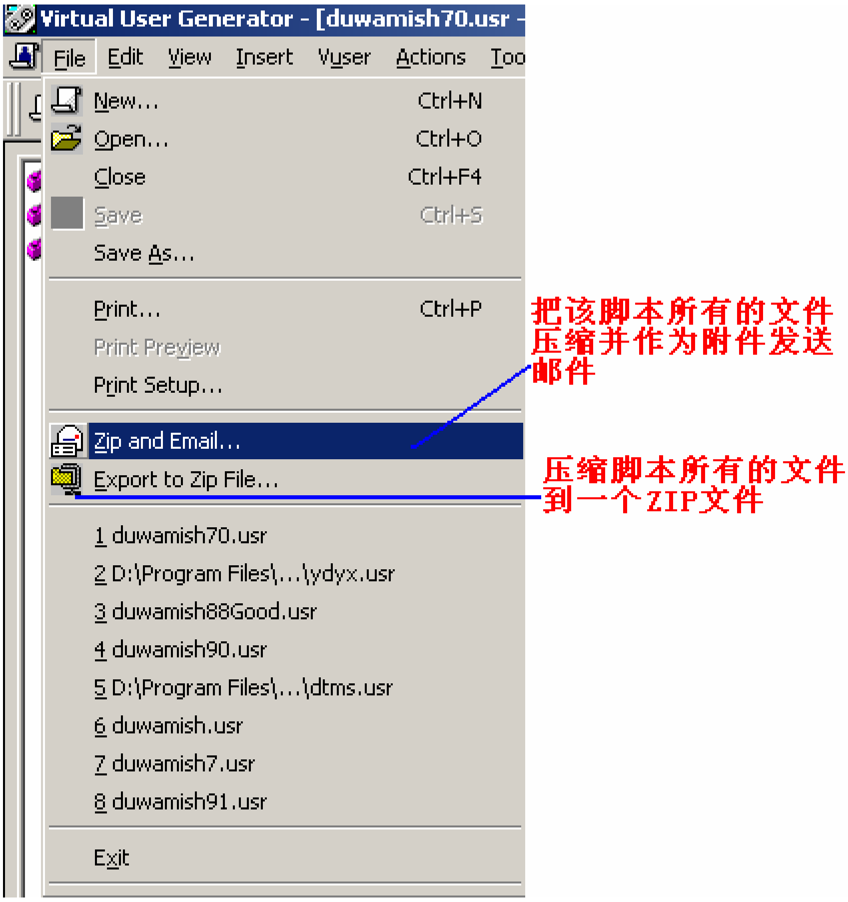
如果编译通过，就会开始运行。然后会出现运行结果。



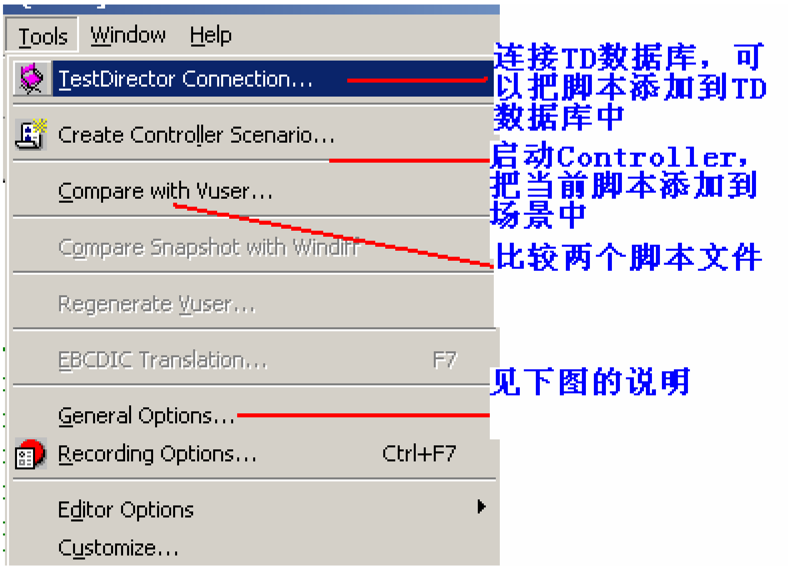
结果和 WinRunner 中格式一样，这里不再多说明。

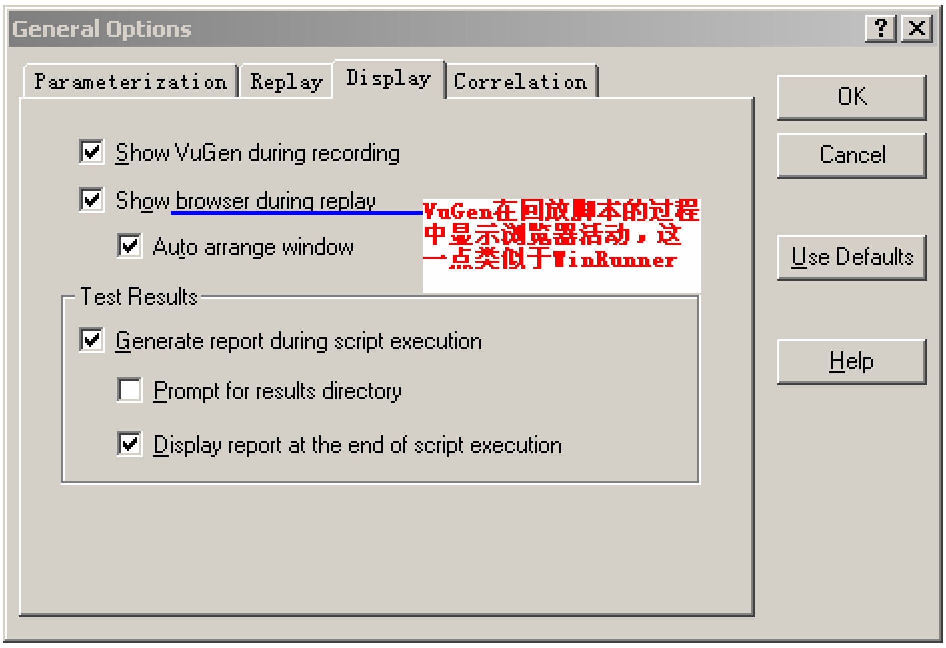
5.5 VuGen 其他有用的功能

5.5.1 压缩脚本文件



5.5.2 tools 菜单



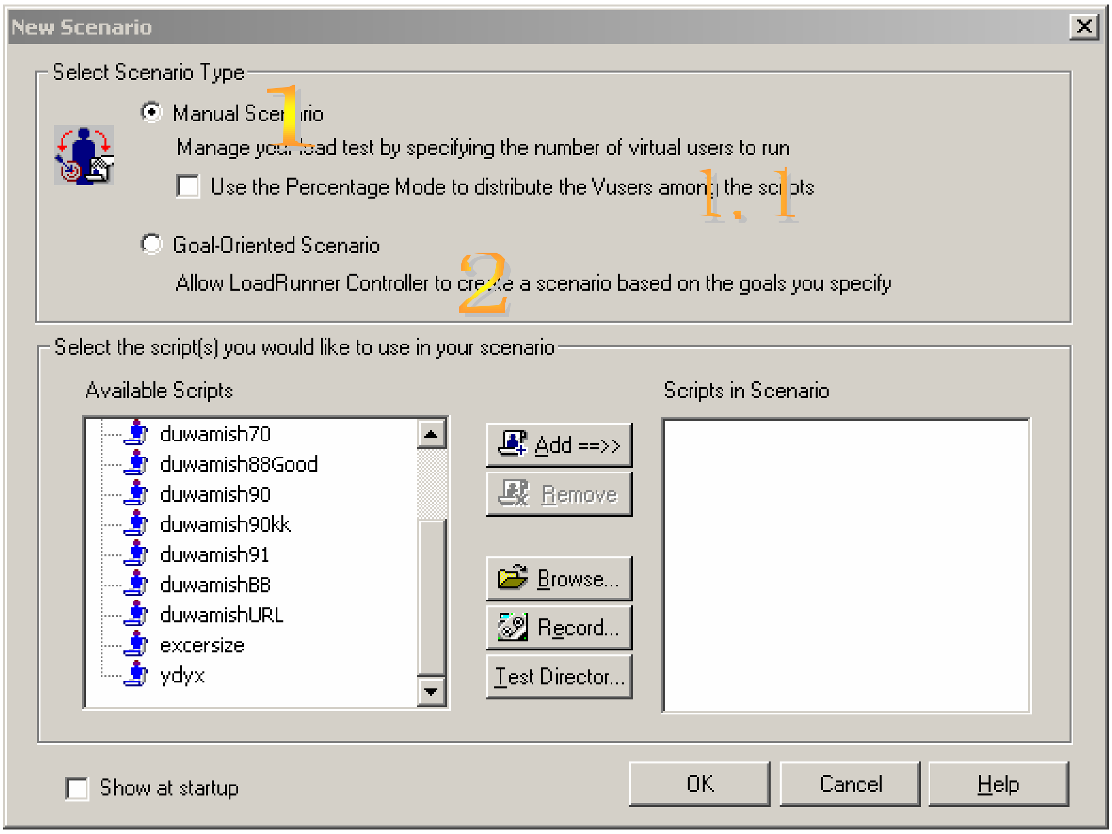


6 创建运行场景

运行场景描述在测试活动中发生的各种事件。一个运行场景包括一个运行虚拟用户活动的 Load Generator 机器列表，一个测试脚本的列表以及大量的虚拟用户和虚拟用户组。

创建运行场景使用 Controller。

在开始菜单中，启动 Controller 程序，出现“New Scenario”窗口。如果没有出现，可以在菜单或者工具栏中点击“New”。



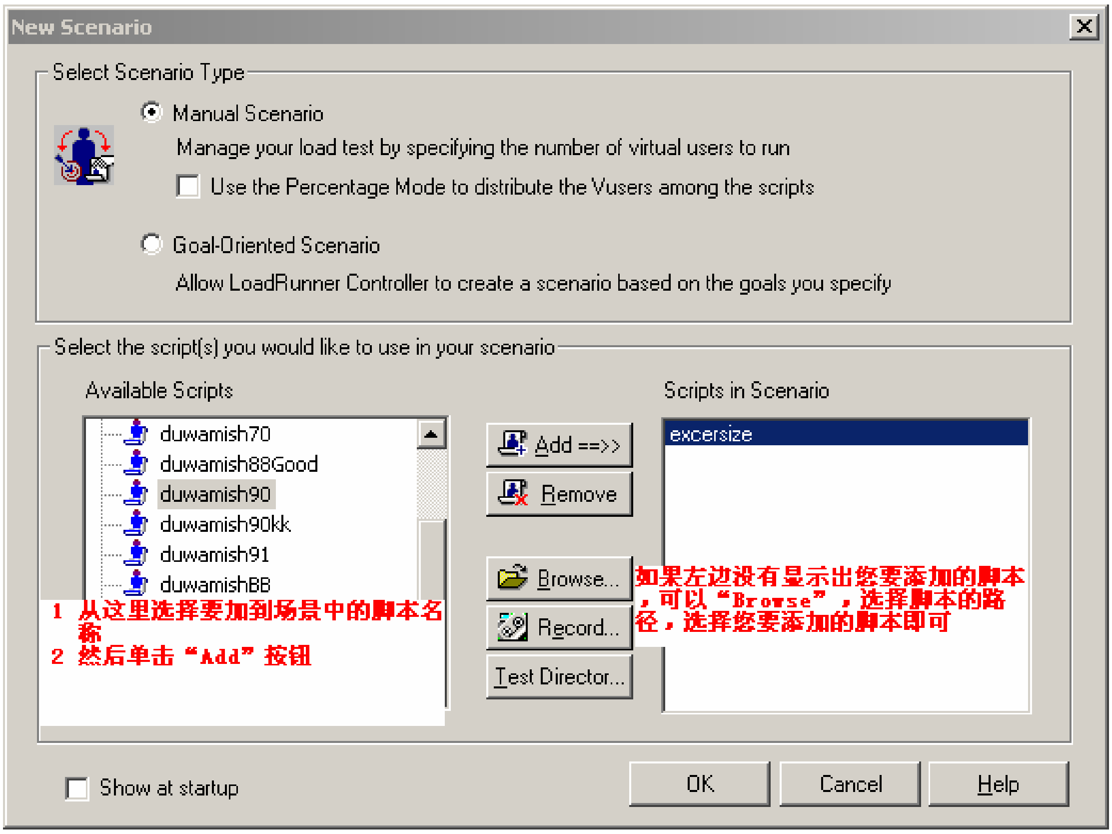
在新建场景的窗口，选择一种场景类型。下面对三种类型进行简单的说明。

- 1　Manual Scenario：该项要完全手动的设置场景。更加详细的信息，请参考 6.1。
- 1.1　Manual Scenario with Percentage Mode：该项只有在“Manual Scenario”选中的情况下才能选择。选择该项后，在场景中我们需要定义要使用的虚拟用户的总数，Load Generator machine 机器集，然后我们为每一个脚本分配要运行的虚拟用户的百分比。更加详细的信息，请参考第 6.2 章。
- 2　Goal—Oriented Scenario：在测试计划中，一般都包括性能测试要达到的目标。选择该项后，LoadRunner 基于这个目标，自动为你创建一个场景。在场景中，我们只要定义好我们的目标即可。更加详细的信息，请参考第 6.3 章。

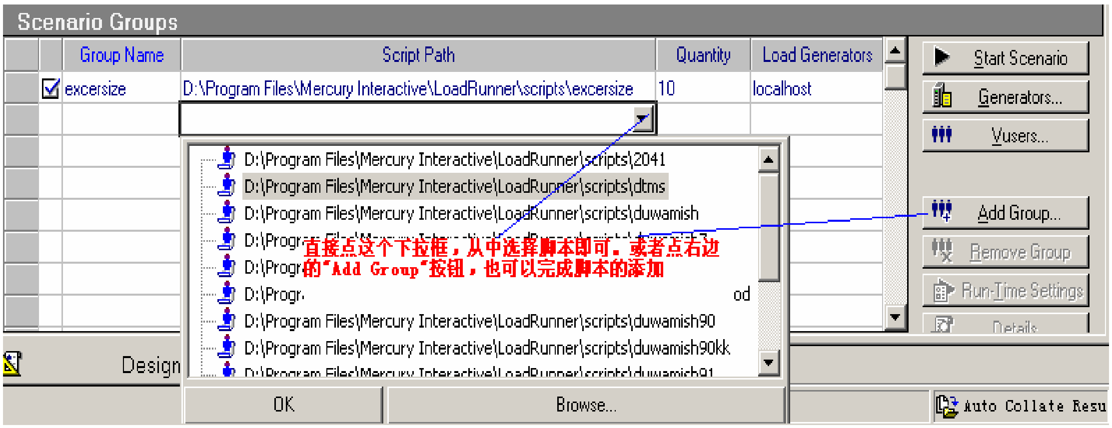
6.1 选择场景类型为 Manual Scenario

6.1.1 选择 Vuser Groups

在上图中，把脚本添加到场景操作很简单。

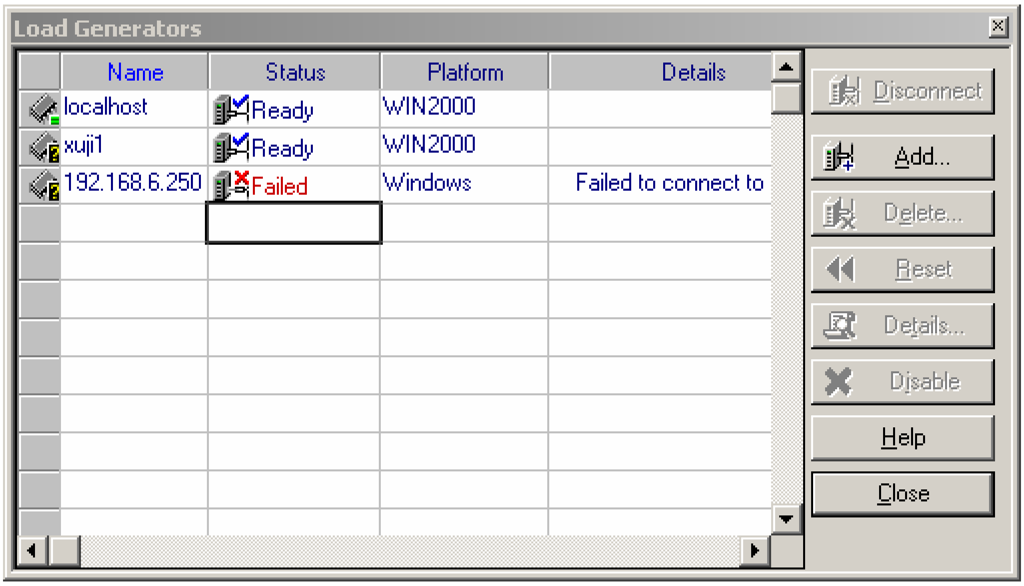


如果在已经打开的场景中，添加脚本



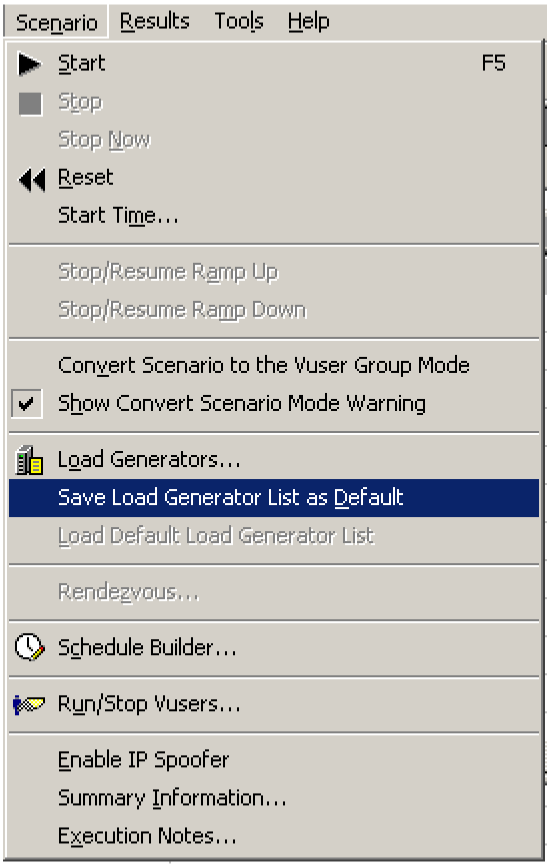
6.1.2 添加 Load Generator Machines

点右边的“Generators”按钮，出现 Load Generators 窗口



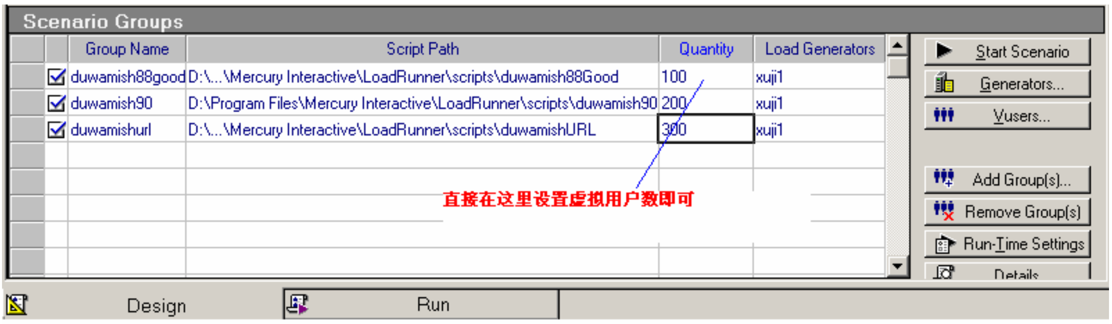
添加 LoadGenerator 后，执行“Connect”操作，使 Status 为 Ready，表示该机器联接正常，如果为 Failed，表示该机器不能联接，请检查原因。

可以把这个列表保存下来，执行菜单命令即可。



6.1.3 添加虚拟用户

首先设置虚拟用户总数。

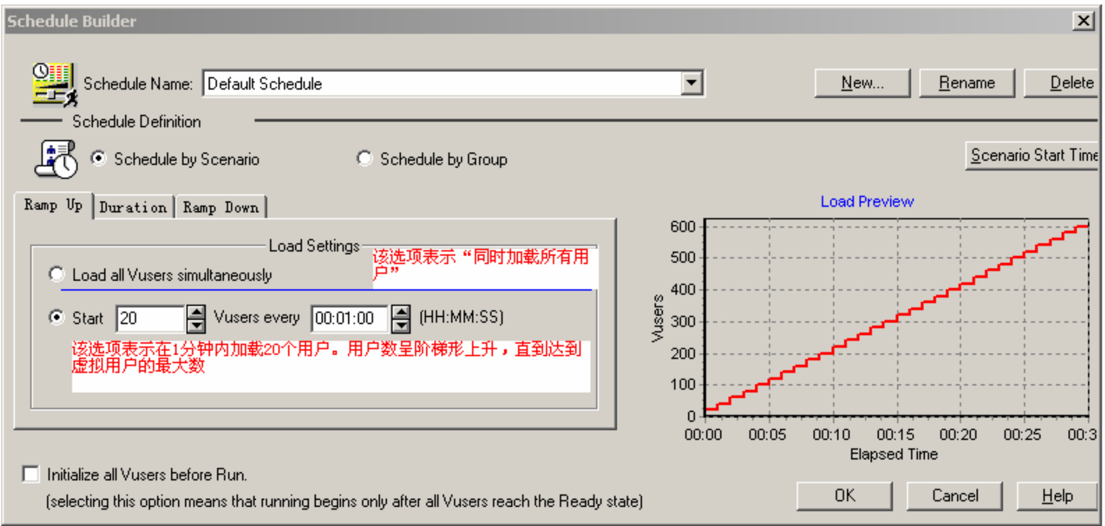


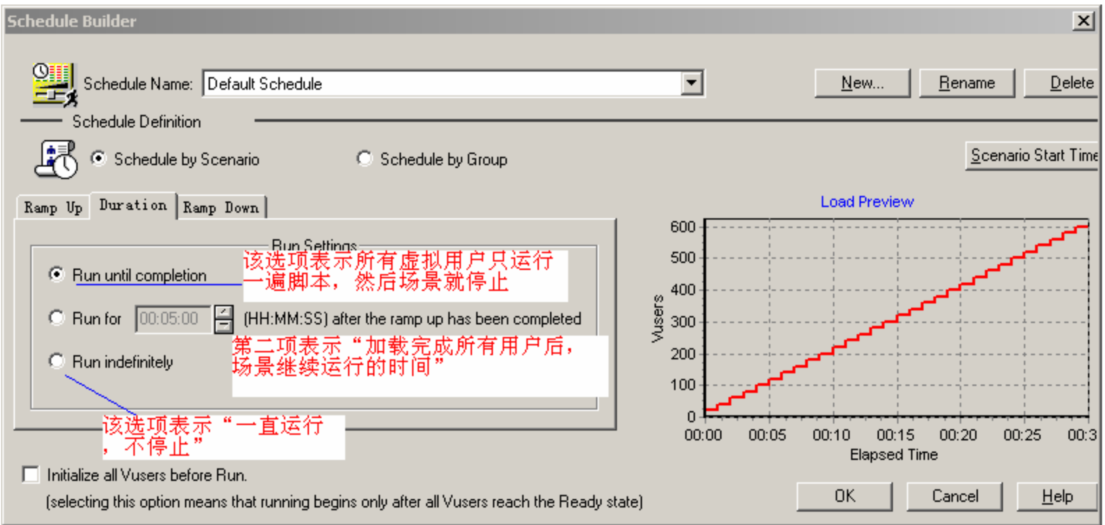
点右边的“VUsers”按钮即可设置，该虚拟用户将在那个 Load Generators 上运行。

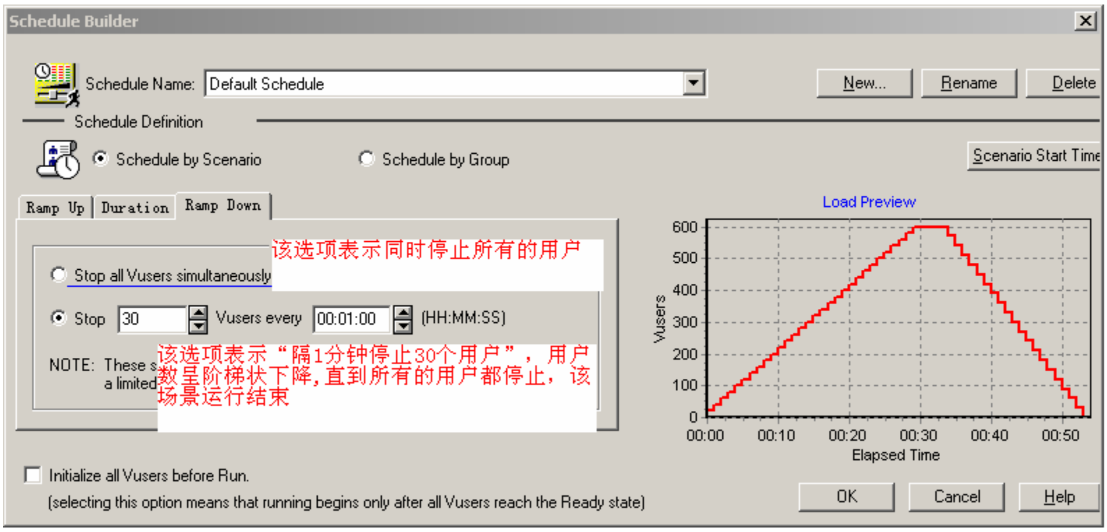
6.1.4 设置 Schedule

这里的设置是非常重要的，也是三种场景类型最重要的区别之处。点 

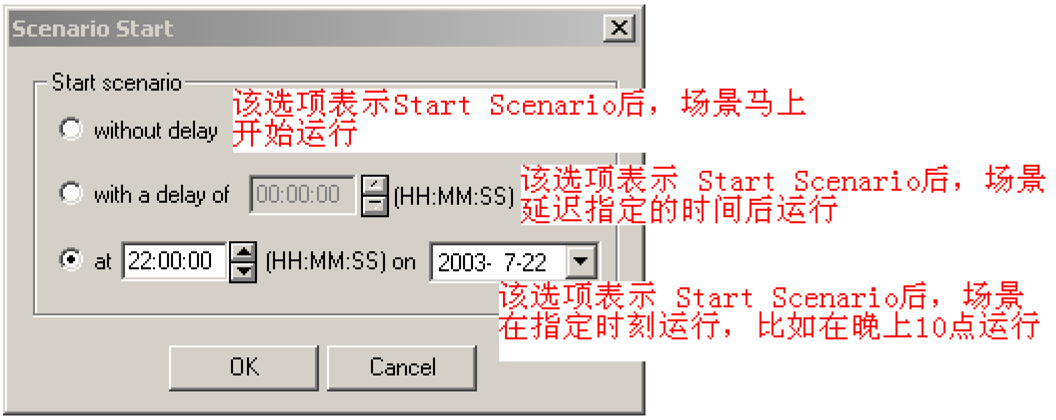
按钮，即可进入 Schedule 设置窗口。





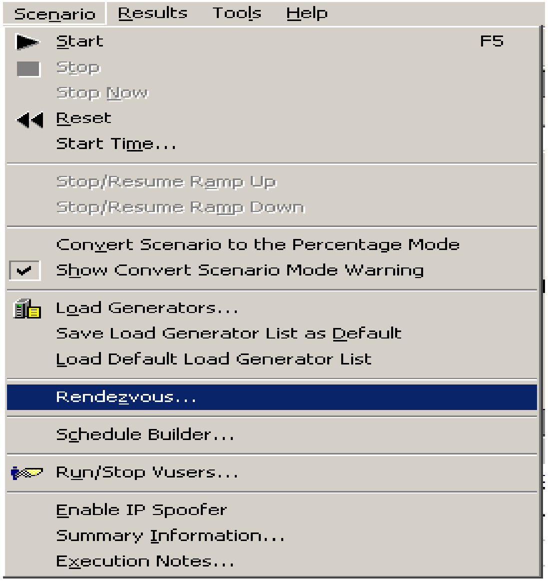


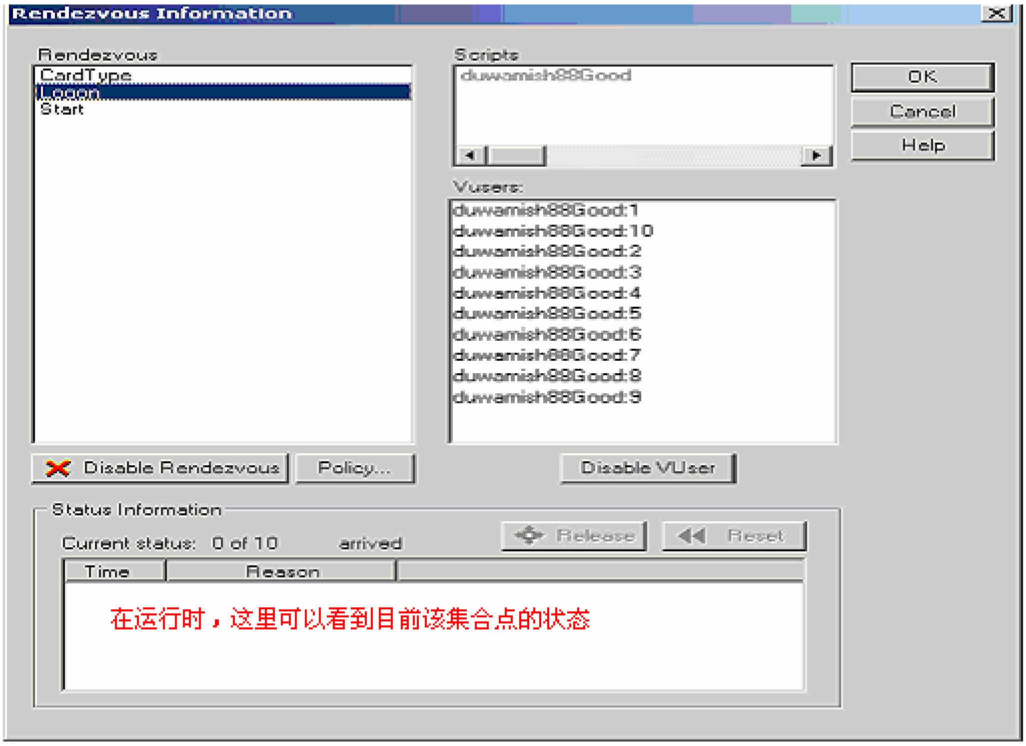
点按钮，进入 Scenario Start Time 窗口



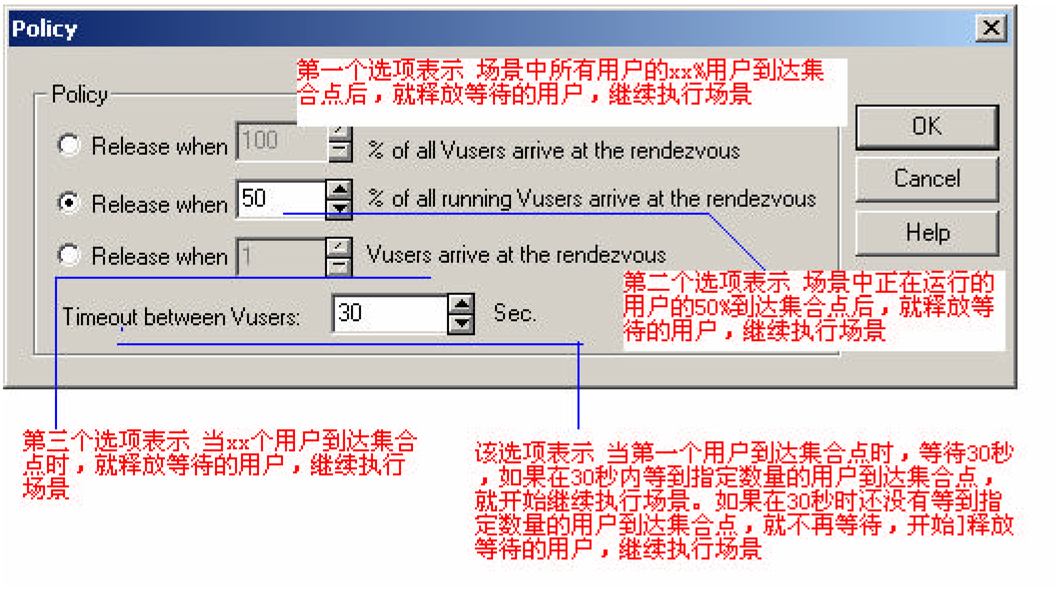
6.1.5 设置集合点

如果在脚本中设置了集合点，还需要在 Controller 中设置集合点策略。
在菜单中调出 设置集合点策略的窗口.



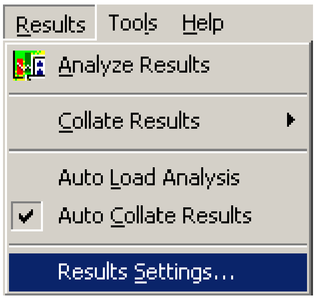


点按钮，进入策略设置窗口

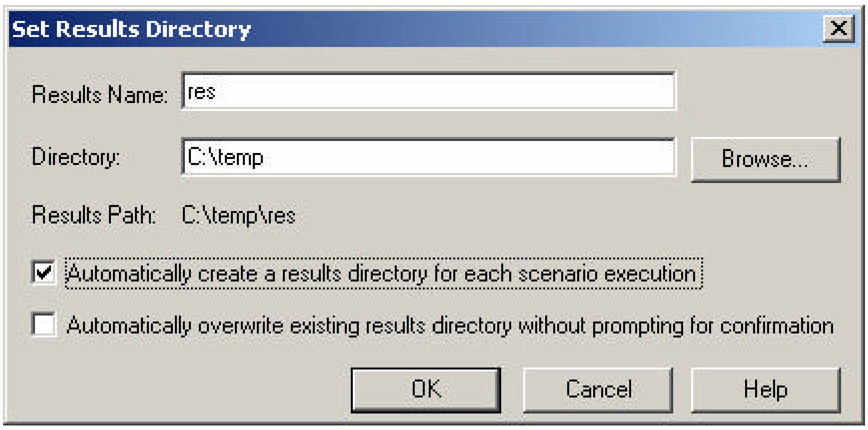


6.1.6 设置结果文件保存路径

通过菜单操作



调出结果文件的保存路径



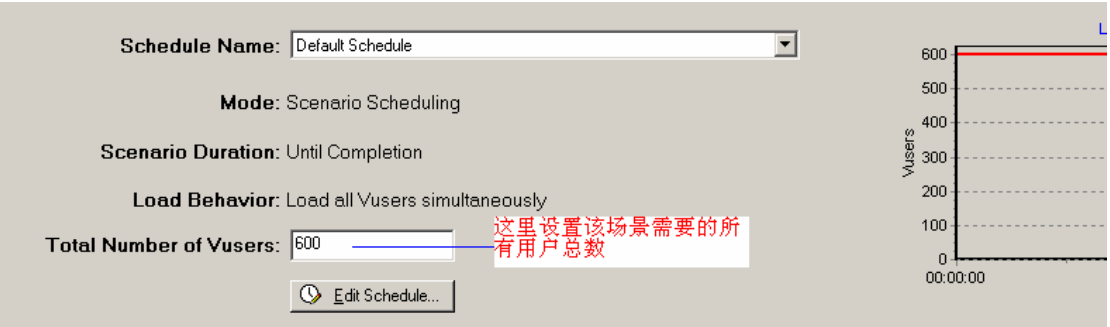
该路径最好在每次场景运行前重新设置一下。

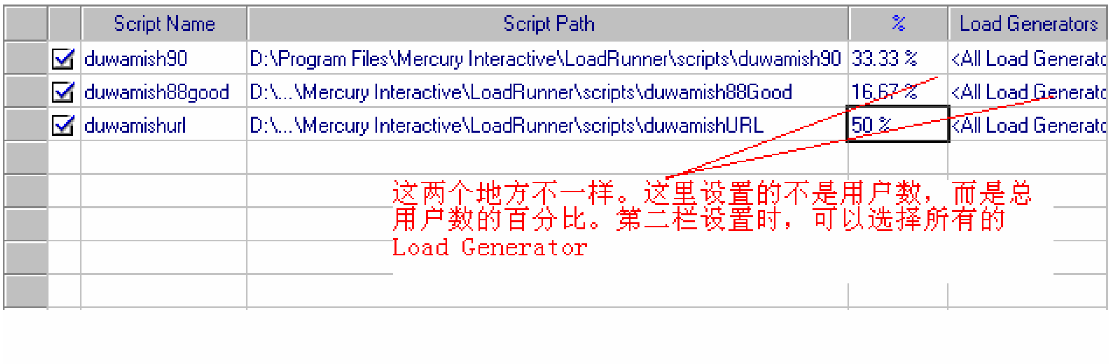
6.1.7 Run-Time Setting

请参考 5.3

6.2 选择场景类型为 Manual Scenario with Percentage Mode

该场景类型和“ Manual Scenario ”类型非常类似，下面简单的对他们不一样的地方进行设置。

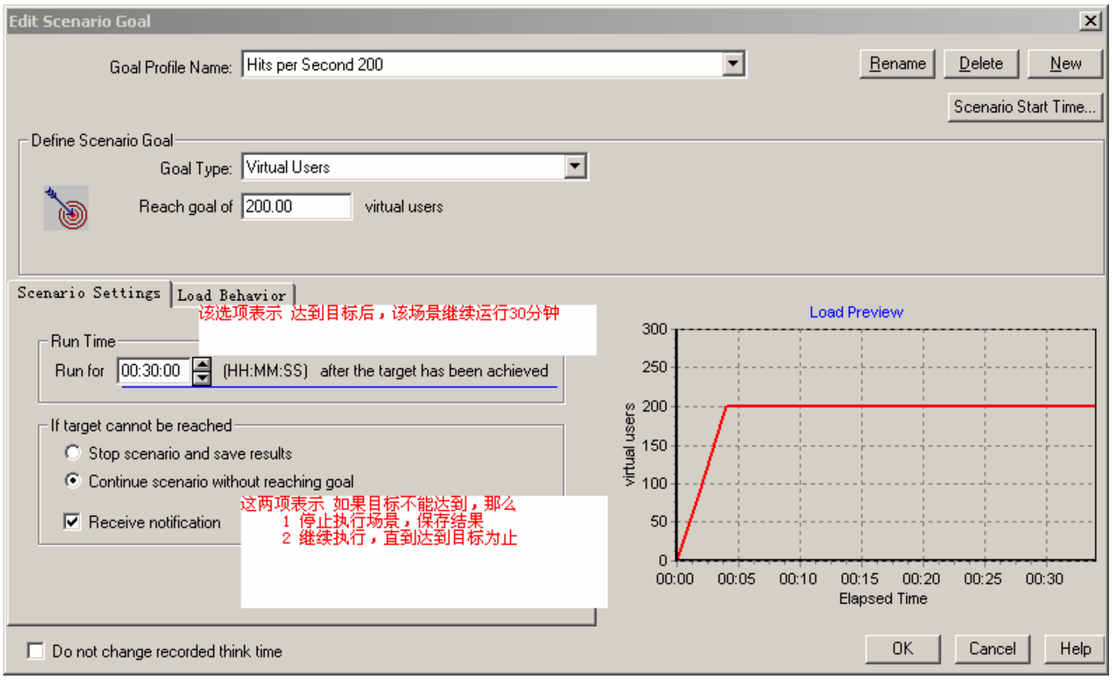


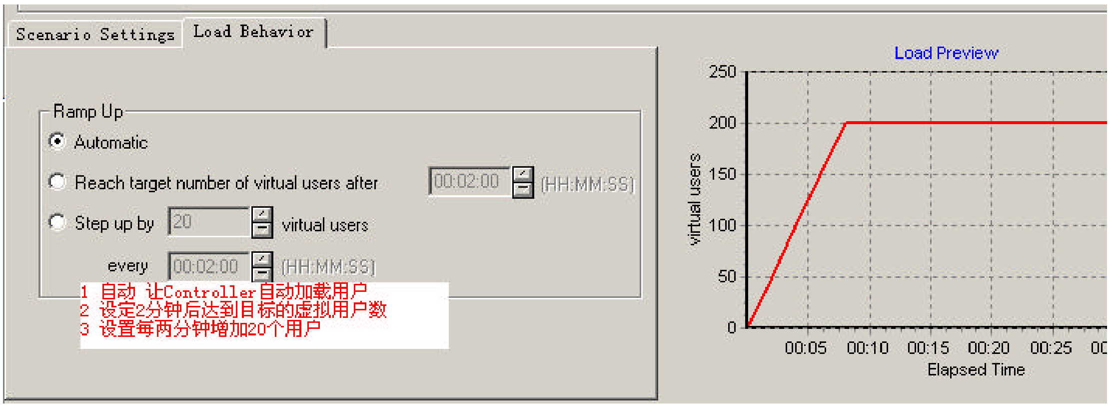


6.3 选择场景类型为 Goal—Oriented Scenario

同样，只对不同的地方进行设置讲解。

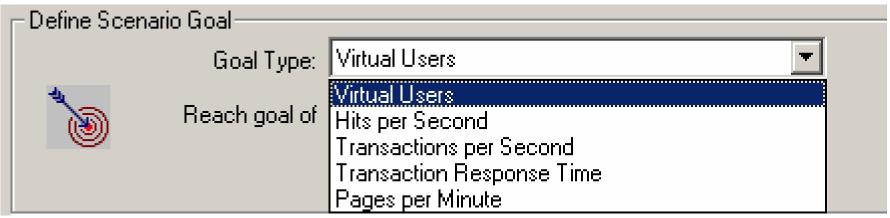
点**按钮，编辑该场景的目标。**





注：以上的说明是以选择的目标为 Virtual Users 时为基础的。选择不同的目标，内容会稍微有一点不同。

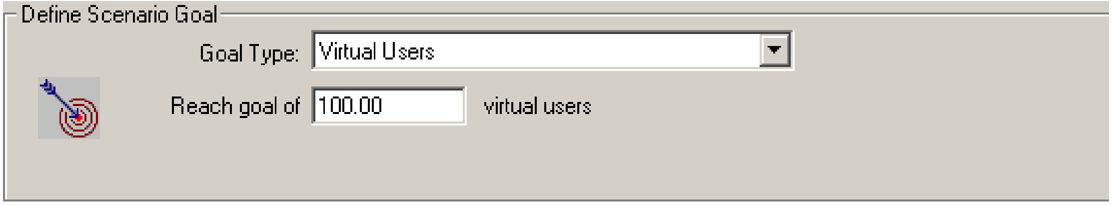
下面重点说一下目标的种类。每次场景运行只能设置一个目标。



6.3.1 Virtual Users Goal

如果需要测试多少人可以同时运行 Web 应用，那么推荐定义 Virtual Users Goal。运行定义该目标类型的场景和运行 Manual 类型的场景类似。

具体的定义方法很简单，不再详细的说明。



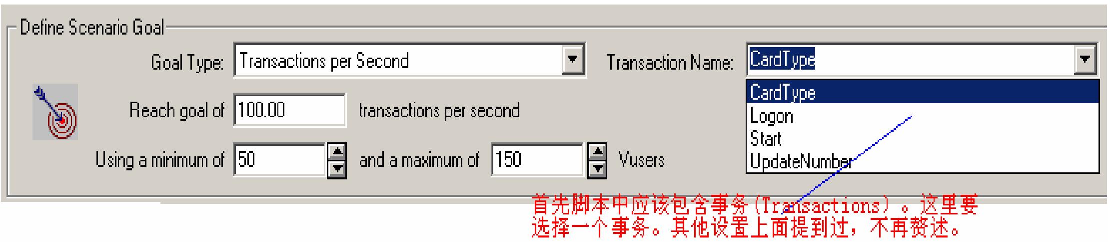
6.3.2 Hits per Second

如果想测试 Web Server 的真正实力，推荐定义目标类型为：Hits per Second、Pages per Minute 或者 Transactions per Second，这些类型都需要指定一个虚拟用户的最小值和最大值的范围。

Controller 试图使用最少的虚拟用户来达到定义的目标。如果使用最少的用户，不能达到目标，Controller 增加用户数，直到定义的最大值。如果使用了最多的虚拟用户数，定义的目标还没有实现，那么需要增加最大用户数，重新执行场景。



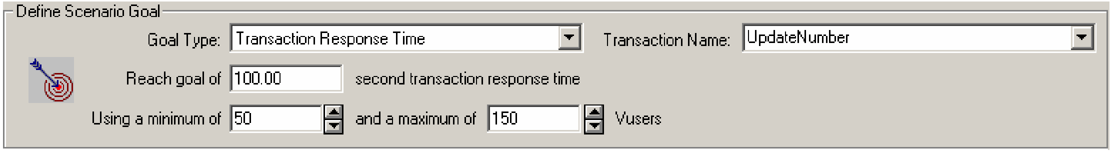
6.3.3 Transactions per Second



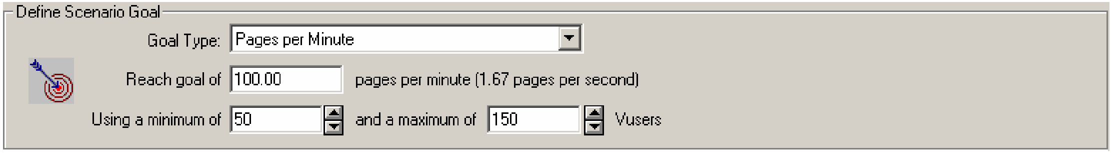
6.3.4 Transactions Response Time

如果想知道在多少用户并发访问网站时，事务的响应时间达到性能指标说明书中规定响应时间的最大值，那么推荐使用 Transactions Response Time 类型。指定需要测试的事务的名称，虚拟用户数量的最小值和最大值，还有预先定义好的事务的响应时间。

在场景运行中，如果使用了最多的虚拟用户，还不能达到定义的最大响应时间，说明 Web Server 还有能力接纳定义的虚拟用户的最多数量；如果在使用了部分虚拟用户，就达到了定义的最大的响应时间，或者 LoadRunner 提示如果使用最多数量的虚拟用户时将要超过最大响应时间，那么需要重新设计或者修补应用程序，同时可能需要升级 Web Server 的软硬件。



6.3.5 Pages per Minute



6.3.6 理解各种类型

如果你定义的类型是 Pages per Minute、Hits/Transactions per Second，Controller 首先用最小用户数除以定义的目标，得到一个值，然后确定每个用户应该达到的 hits/transactions 或者 pages per minute，然后 controller 开始按照以下的策略加载用户：

- 如果选择的是自动的加载虚拟用户，LoadRunner 会首先加载 50 个用户。如果定义的最大用户数小于 50，LoadRunner 就会一次加载所有的虚拟用户。
- 如果选择的是在场景运行一段时间后达到目标，LoadRunner 就会尝试在定义的这段时间内达到目标，根据时间限制和计算出的每个用户的 hits、transactions 或者 pages，LoadRunner 确定第一批加载多少用户。
- 如果选择的是按照一定的阶段达到目标（也就是先在 x 长时间内达到 y pages/hits，然后再达到下一个目标），LoadRunner 计算每个用户应该达到的数字后，再确定第一批加载多少用户。

每加载一批用户后，LoadRunner 会判断是否达到这批用户的目标。如果这批用户的目标没有达到，LoadRunner 重新计算每一个用户应该达到的目标数字后，重新调整下一批加载用户的数量。默认情况下，LoadRunner 每两分钟加载一批用户。

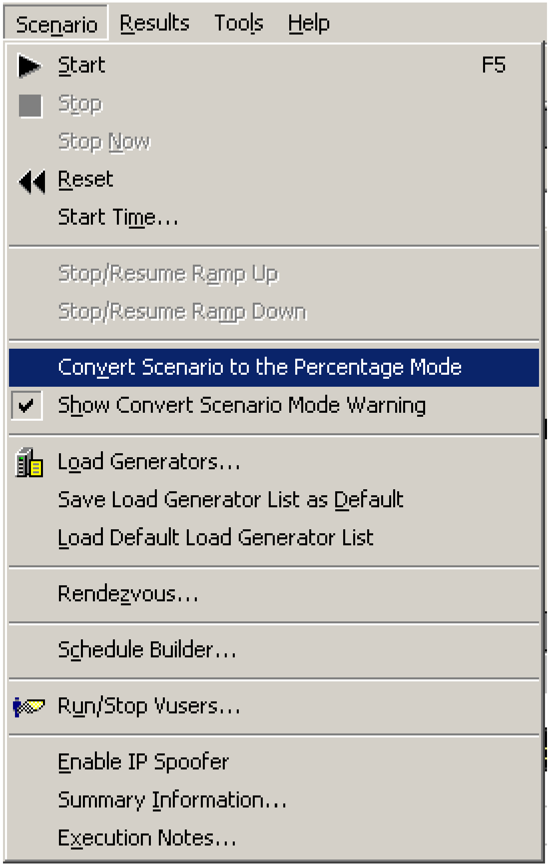
如果 Controller 加载了最多数量的用户还没有达到预定的目标，LoadRunner 会重新计算每个用户的目标，然后同时运行最大数量的用户，尝试达到预定的目标。

如果出现以下情况，Pages per Minute、Hits/Transactions per Second 类型的场景会置于“Failed”状态：

- Controller 使用了指定的最大数量的用户，并且两次都没有达到目标
- 所有的用户运行都失败
- 没有足够的 Load Generators 机器（现有的机器已经超载运行的情况下）
- Controller 增加了几批用户后，pages per minute 或者 hits/transactions per second 没有增加
- Controlller 加载第一批用户后，定义的目标没有被捕捉到

6.4 其他有用的设置

6.4.1 场景类型的转化



使用这个选项，可以在 Percentage Mode 和 Vusers Group 之间互相转化，不过一些设置可能会丢失。更详细的信息请参考帮助文档。

6.4.2 启用 IP Spoofer（IP 欺骗）

当运行场景时，虚拟用户使用它们所在的 Load Generator 的固定的 IP 地址。同时每个 Load Generator 上运行大量的虚拟用户，这样就造成了大量的用户使用同一 IP 同时访问一个网站的情况，这种情况和实际运行的情况不符，并且有一些网站会根据用户 IP 来分配资源，这些网站会限制同一个 IP 的登陆，使用等等。为了更加真实的模拟实际情况，LoadRunner 允许运行的虚拟用户使用不同的 IP 访问统一网站，这种技术称为"IP 欺骗"。

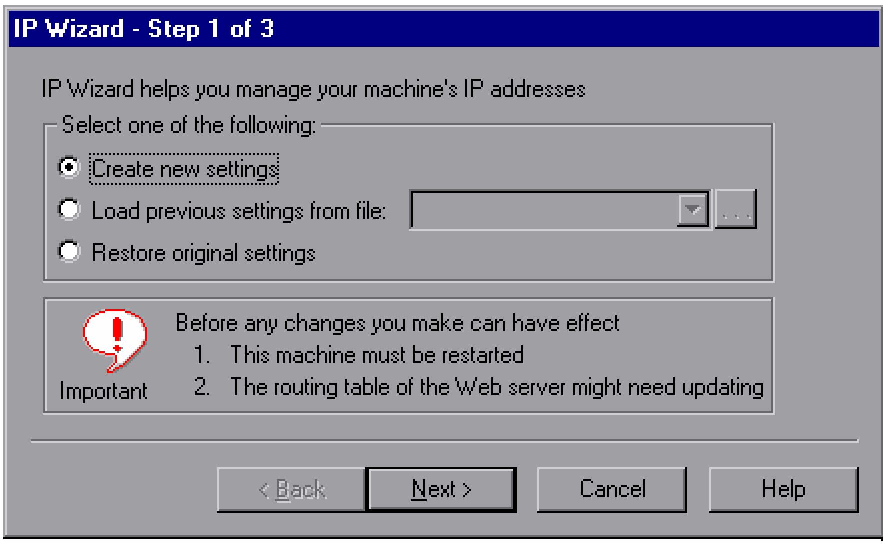
启用该选项后，场景中运行的虚拟用户将模拟从不同的 IP 地址发送请求。该选项非常的有用。注意：IP Spoofer 在连接 Load Generators 之前启用。

要使用 IP 欺骗，各个 Load Generator 机器必须使用固定的 IP，不能使用动态 IP（即 DHCP）。

使用 IP Spoofer 的步骤如下：

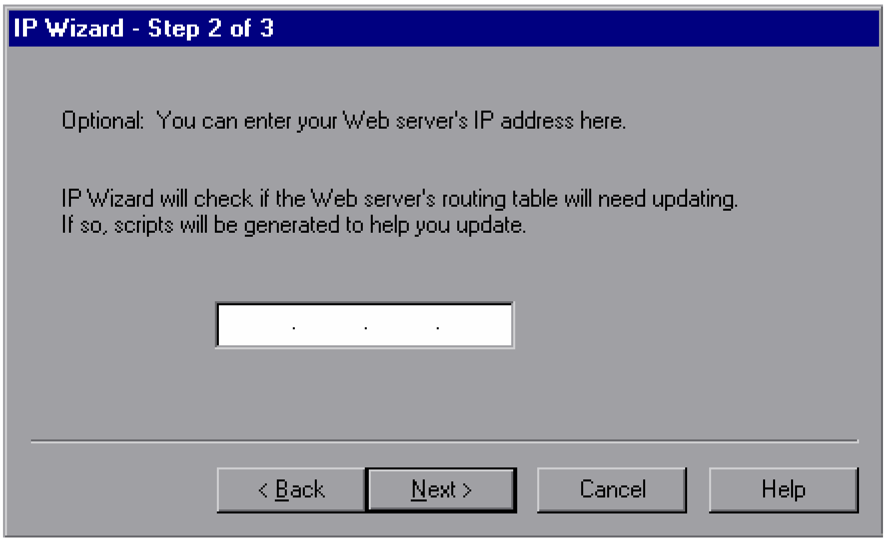
1. **使用 IP Wizard：在"开始"菜单程序中，找到 LoadRunner——Tools——IP Wizard，运行它**

注意：运行 IP Wizard 程序的机器必须使用固定的 IP，不能使用动态 IP



第一次运行 IP Wizard 需要选择第一项"Create new settings"，如果以前运行过，可以选择第二项"Load previous settings from file"，选择保存好的文件；第三项用于使用 IP 欺骗进行测试完成后，释放 IP 的过程（因为该机会占用大量的 IP 资源，可能会导致其他机器没有 IP 可用的尴尬局面，使用该项，可以恢复到原来的状况）。

这里选择第一项，"Next"，出现 IP Wizard 的第二个窗口

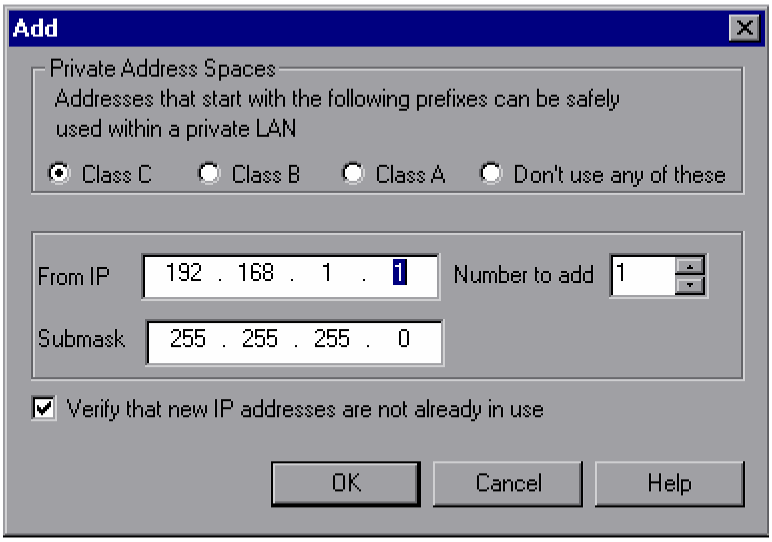


这里输入 Web Server 的 IP 地址，然后 Next，出现向导的第三个窗口。

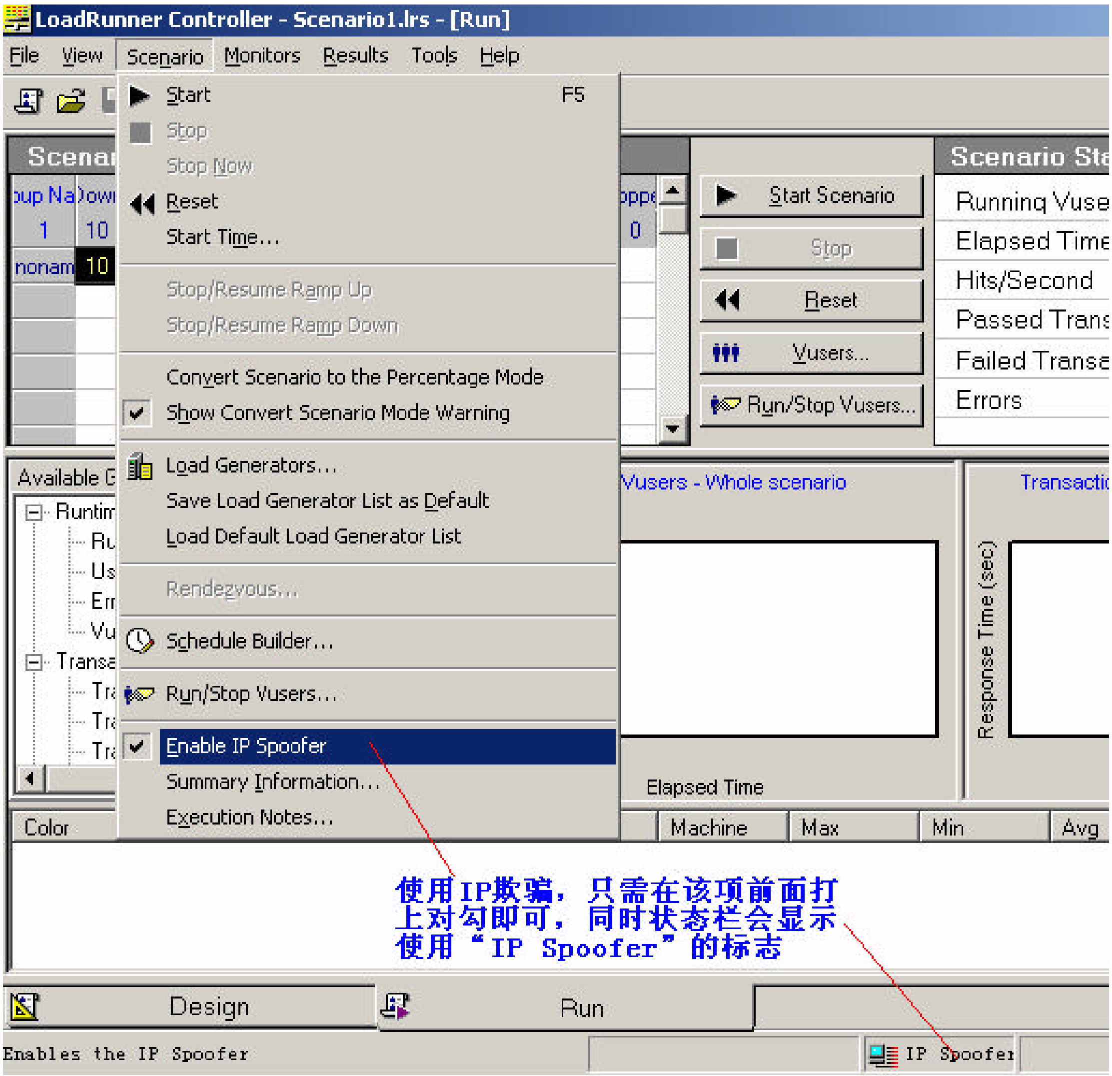
从"From IP"文本框中输入要使用 IP 范围的第一个 IP 值，然后在"Numbers to Add"输入一个数字，表示 IP 范围的值；假如第一个 IP 为 192.168.6.100，范围大小为 100，那么 IP Wizard 将会使用 192.168.6.N（100<=N<200），当然这个范围内已经使用的 IP 地址除外，否则会引起 IP 冲突。"Submask"采用默认情况即可，取决于使用的那种类型的网络 IP，一般局域网内采用 Class C 即可。

然后"OK"，然后 IP Wizard 开始检查该范围内没有使用的 IP，并把没有使用的 IP 添加到本机的 IP 窗口中。到最后一个窗口，直接点"Finish"，

使用 IP Wizard 后，最后重新启动机器。

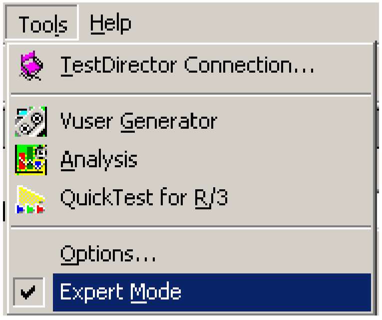


2.　在 Controller 的场景中，启用 IP Spoofer 即可。启用方法很简单。

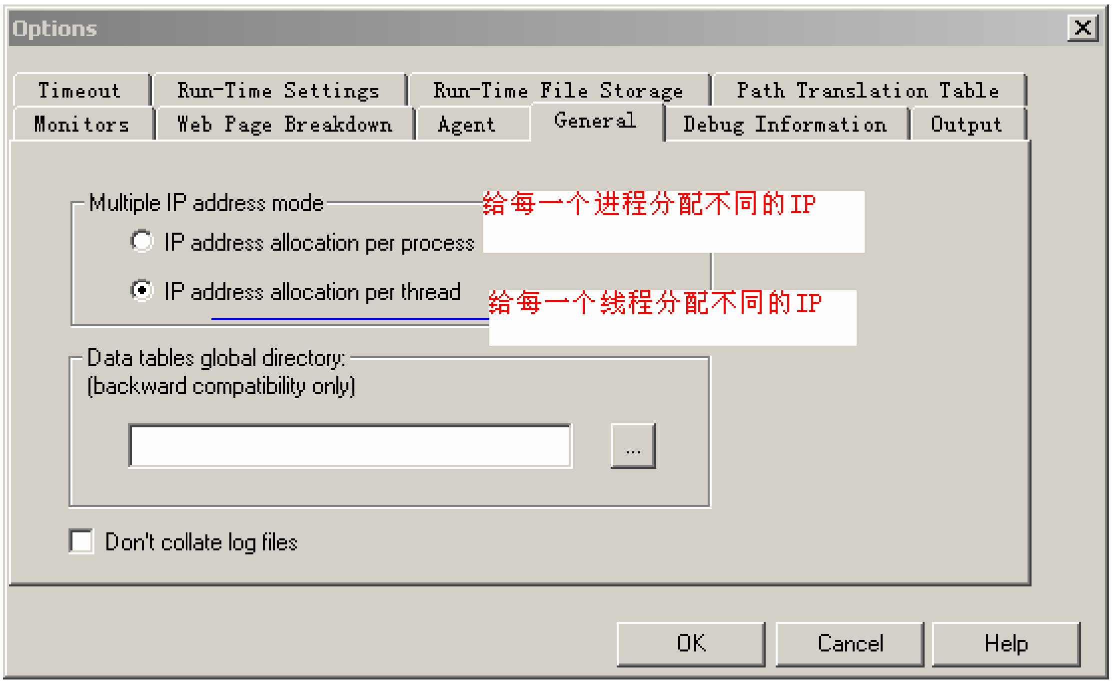


6.4.3 Options 设置

首先使用 Export Mode（专家模式）



然后选择上述菜单中的 Options



由于我们在 Controlller 的 Run-Time Setting 中设置了以多线程方式运行，所以这里我们应该选择第二项，为每一个线程分配不同的 IP。

其他设置采用默认即可，这里不再详细说明。

6.4.4 优化 Controller 和 Load Generators 计算机

如果控制机（Controller machine）和 Load Generators 计算机运行的都是 Windows2000，那么下面两个简单的技巧可以提高性能

- 在 Load Generators 计算机上，依次进入"控制面板"——"系统"——选择"高级"标签页，点"性能选项"按钮，选择优化"后台服务"选项，这样可以提高性能，从而可以在每个 Load Generators 上运行更多的虚拟用户
- 在 Controller 计算机上，按照以上的步骤，进入"性能选项"窗口，不过这里选择优化"应用程序"

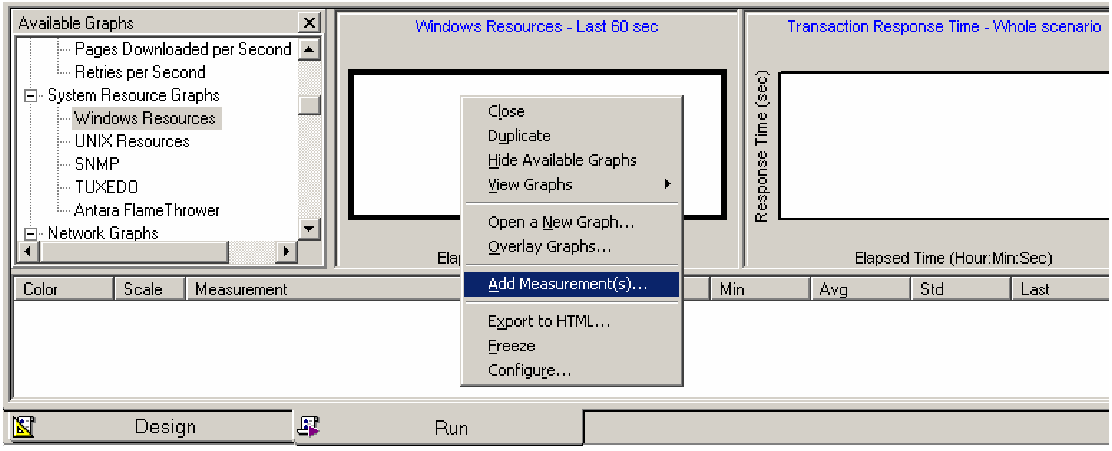
7 分析以及监视场景

在运行过程中，可以监视各个服务器的运行情况（DataBase Server、Web Server 等）。监视场景通过添加性能计数器来实现。**这一章非常的重要，确定系统瓶颈全靠它了。**

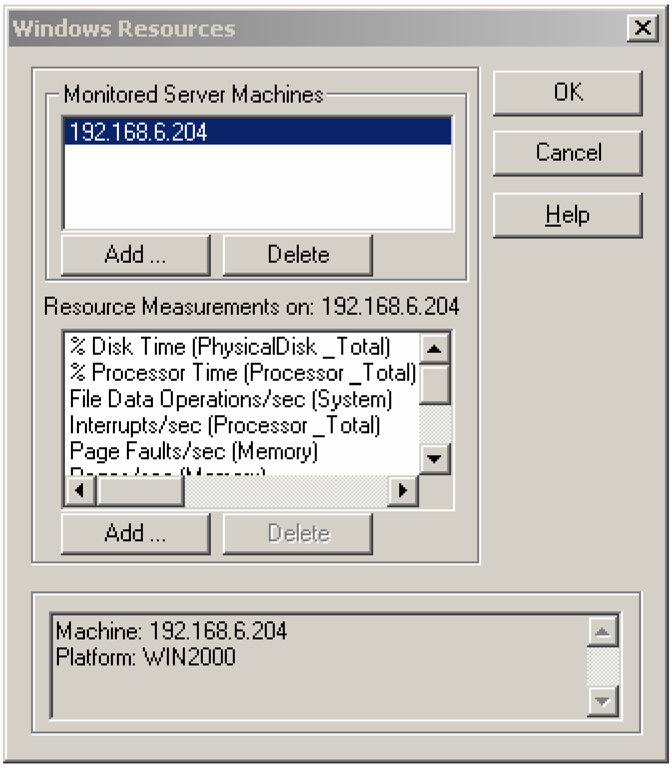
下面重点讲讲需要添加那些计数器，以及那些计数器代表什么意思。

由于 Win2000 Professional、Server 以及 Advanced Server 提供的计数器不完全相同，这里我们讨论将以 Server 为基准。

监视场景需要在 Run 视图中设置



然后，出现添加计数器的对话框



其他的操作就和控制面板“性能”中添加性能计数器的操作一样，这里不再详细说明。

本章主要说明一下各个系统计数器的含义（数据库的计数器不做重点，只是拿 SQL Server2000 作为例子进行说明。因为数据库各个版本之间差异比较大，请参考您使用的数据库系统的帮助）。

7.1 Memory 相关

内存是第一个监视对象，确定系统瓶颈的第一个步骤就是排除内存问题。内存短缺的问题可能会引起各种各样的问题。

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
Memory	Available MBytes	物理内存的可用数（单位 Mbytes）。默认情况下 IIS5.0 使用 50%的可用物理内存，作为 IIS 的文件缓存（file cache）。IIS 基本占用 2.5 MB，每个附加连接将在此基础上占用 10 KB 左右。	至少要有 10% 的物理内存值
Memory	Page/sec Page Faults/sec Pages Input/sec Page Reads/sec Transition Faults/sec	当处理器向内存指定的位置请求一页（可能是数据或代码）出现错误时，这就构成一个 Page Fault。如果该页在内存的其他位置，该错误被称为软错误（用 Transition Fault/sec 计数器衡量）；如果该页必须从硬盘上重新读取时，被称为硬错误。许多处理器可以在有大量软错误的情况下继续操作。但是，硬错误可以导致明显的拖延。Page Faults/sec 是处理器每秒钟处理的错误页（包括软错误和硬错误）。 Pages Input/sec 是为了解决硬错误页，从硬盘上读取的页数，而 Page Reads/sec 是为了解决硬错误，从硬盘读取的次数。如果 Page Reads/Sec 比率持续保持为 5，表示可能内存不足。 Pages/sec 是指为解析硬页错误从磁盘读取或写入磁盘的页数。	Page/sec 推荐 00-20（如果服务器没有足够的内存处理其工作负荷，此数值将一直很高。如果大于 80，表示有问题）。 这些计数器的值比较低，说明 Web 服务器响应请求比较快，否则可能是服务器系统内存短缺引起（也可能是缓存太大，导致系统内存太少）。Page Input/sec 的值可以衡量出硬错误页发生的速率，通常它的值会大于或者等于 Page Reads/sec。
Memory	Cache Bytes	文件系统缓存（File System Cache），默	默认情况下

		认情况下为 50%的可用物理内存。如 IIS5.0 运行内存不够时，它会自动整理缓存。需要关注该计数器的趋势变化。	为 50%的可用物理内存
Internet Information Services Global	File Cache Hits % File Cache Flushes File Cache Hits	File Cache Hits %是文件缓存命中全部缓存需求的比例，反映了 IIS 的文件缓存设置的工作情况。而 File Cache Hits 是文件缓存命中的具体值，File Cache Flushes 是自服务器启动之后文件缓存刷新次数，如果刷新太慢，会浪费内存；如果刷新太快，缓存中的对象会太频繁的丢弃生成，起不到缓存的作用。通过 File Cache Hits 和 File Cache Flushes 可以得到一个适当的刷新值（参考 IIS 的设置 ObjectTTL、MemCacheSize、MaxCacheFileSize）。	（对于一个大部分是静态网页组成的网站）File Cache Hits% 在 80%左右属于非常好！
Memory	Pool Paged Bytes Pool Nonpaged Bytes	这两个计数器监视服务器上各个进程的分页池字节数和非分页池字节数。	在访问数比较固定的情况下，Pool Nonpaged Bytes 是比较固定的，如果访问数逐步增加，该值会缓慢的增加。
Process	Virtual Bytes(实例 inetinfo、dllhost) Working Set(实例 inetinfo、dllhost) Dllhost#n 进程都要添加计数器	Virtual Bytes 计数器监视 IIS5.0 保留的虚地址空间的数量，实例化为 inetinfo 进程(IIS 运行的核心)和 Dllhost 进程(隔离/连接池 的应用程序必需的)。Working Set 计数器反映了每个进程使用的内存页的数量。系统的内存页(pool Page)只能由操作系统的核心模块直接访问，用户进程不能访问。运行 IIS5.0 的服务器上，负责 web 连接的线程以及它需要的一些对象都保存在未分页的池中(nonpaged pool)，比如文件句柄和 socket 连接	
Process	Private Bytes	指这个处理不能与其他处理共享的、已分配的当前字节数。	
Memory	Committed Bytes	Committed Byte 是指以字节表示的确认虚拟内存。(确认内存是指为磁盘分页文件在磁盘上保留的空间以便在需要将其写回磁盘时使用)。	推荐不超过物理内存的 75%

内存问题主要检查应用程序是否存在内存泄漏。如果发生了内存泄漏，Process\Private Bytes

计数器和 Process\Working Set 计数器的值往往会升高，同时 Available Bytes 的值会降低。内存泄漏应该通过一个长时间的，用来研究分析当所有内存都耗尽时，应用程序反应情况的测试来检验。

7.2 Processor 相关

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
Sytem	Processor Queue Length	Processor Queue Length 是指处理列队中的线程数。即使在有多个处理器 的计算机上处理器时间也会有一个单列队。不象磁盘计数器，这个计数器仅 计数就绪的线程，而不计数运行中的线程。如果处理器列队中总是有两个以上的线程 通常表示处理器堵塞。	小于 2。显示在由 Web 服务器所有处理器共享的队列中等待执行的线程数。处理器瓶颈会导致该值持续大于 2。
Processor	%Processor Time	CPU 使用率。这是查看处理器饱和状况的最佳计数器。显示所有 CPU 的线程处理时间。如果一个或多个处理器的该数值持续超过 90%，则表示此测试的负载对于目前的硬件过于沉重。为多处理器服务器添加该计数器的 0 到 x 个实例。	小于 75%。排除内存因素，如果该计数器的值比较大，而同时网卡和硬盘的值比较低，那么可以确定 CPU 瓶颈
System	Context Switches/sec	Context Switches/sec 指计算机上的所有处理器全都从一个线程转换到另一个线程的综合速率。当正在运行的线程自动放弃处理器时出现上下文转换，由一个有更高优先就绪的线程占先或在用户模式和特权(内核)模式之间转换以使用执行或分系统服务。它是在计算机上的所有处理器上运行的所有线程的 Thread: Context Switches/sec 的总数并且用转换数量衡量。在系统和线程对象上有上下文转换计数器。	如果切换次数到 5000*CPU个数和 10000*CPU 个数中，说明它忙于切换线程而不是处理 ASP 脚本。
Processor	%Privileged Time	% Privileged Time 是在特权模式下处理线程执行代码所花时间的百分比。当调用 Windows 系统服务时，此服务经常在特权模式运行，以便获取对系统专	

		有数据的访问。在用户模式执行的线程无法访问这些数据。 对系统的调用可以是直接的(explicit)或间接的(implicit)，例如页面错误或中断。不像某些早期的操作系统，Windows 除了使用用户和特权模式的传统保护模式之外，还使用处理边界作为分系统保护。某些由 Windows 为您的应用程序所做的操作除了出现在处理的特权时间内，还可能在其他子系统处理出现。	
Thread	Context Switches/sec （实例化 inetinfo 和 dllhost 进程	如果你决定要增加线程字节池的大小，你应该监视这三个计数器（包括上面的一个）。增加线程数可能会增加上下文切换次数，这样性能不会上升反而会下降。如果十个实例的上下文切换值非常高，就应该减小线程字节池的大小。	
Processor	Interrupts/sec %DPC Time	这两个计数器能够反映处理器用在处理中断以及推迟处理调用的时间。Interrupts/sec 指处理器每秒钟接收并维护的硬件中断的平均值。正常的线程操作在中断时悬停。大多数的系统时钟每隔 10 毫秒中断处理器一次，形成了间隔活动的后台。	如果处理器使用率超过 90% 且 % Interrupt Time 大于 15%，则处理器可能负荷过重，并发生中断。

判断应用程序是否存在处理器瓶颈的方法：如果 Processor Queue Length 显示的队列长度保持不变（>=2）个并且处理器的利用率%Processor Time 超过 90%，那么很有可能存在处理器瓶颈。

如果发现 Processor Queue Length 显示的队列长度超过 2，而处理器的利用率却一直很低，那么或许更应该去解决处理器阻塞问题，这里处理器一般不是瓶颈。

如果系统由于应用程序代码效率低下或者系统结构设计有缺陷而导致大量的上下文切换（Context Switches/sec 显示的上下文切换次数比较大），那么就会占用大量的系统资源。如果系统的吞吐量降低并且 CPU 的使用率很高，并且此现象发生时切换水平在 15000 以上，那么意味着上下文切换次数过高

同时还可以比较 Context Switches/sec 和%Privileged Time 来判断上下文切换是否过量。如果后者的值超过 40%，且上下文切换的速率也很高，那么应该检查为什么会产生这样高的上下文切换。

7.3 网络吞吐量以及带宽

Object（对象）	Counters（计数器名称）	Description（描述）	参考值

Network Interface	Bytes Total/sec	Bytes Total/sec 为发送和接收字节的速率，包括帧字符在内。判断网络连接速度是否是瓶颈，可以用该计数器的值和目前网络的带宽比较	该计数器的值和目前网络的带宽相除，结果应该小于 50%
Web Service	Maximum Connections、Total Connection Attempts	Maximum Connections ：“最大连接数”是和 Web 服务同时建立起的最大连接数。 Total Connection Attempts ：“连接尝试总数”是从服务启动时利用 Web 服务尝试连接的总数。该计数器应用于全部所列的实例。	

7.4 磁盘相关

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
Network Interface	Bytes Total/sec	Bytes Total/sec 为发送和接收字节的速率，包括帧字符在内。判断网络连接速度是否是瓶颈，可以用该计数器的值和目前网络的带宽比较	
Processor	%Processor Time % Privileged Time	CPU 使用率 该计数器对应于处理器执行 Windows® 2000 内核命令（如处理 SQL Server I/O 请求）所用时间的百分比。如果 Physical Disk 计数器的值很高时该计数器的值也一直很高，则考虑使用速度更快或效率更高的磁盘子系统。	
PhysicalDisk	%Disk Time	% Disk Time 指所选磁盘驱动器忙于为读或写入请求提供服务所用的时间 的百分比。如果三个计数器都比较大，那么硬盘不是瓶颈。如果只有%Disk Time 比较大，另外两个都比较适中，硬盘可能会是瓶颈。在记录该计数器之前，请在 Windows 2000 的命令行窗口中运行 diskperf -yD。若数值持续超过 80%，则可能是内存泄漏。	
PhysicalDisk	Average Disk Queue Length	指读取和写入请求(为所选磁盘在实例间隔中列队的)的平均数。	
PhysicalDisk	Average Disk Read Queue Length	指读取请求(为所选磁盘在实例间隔中列队的)的平均数。	

PhysicalDisk	Average Disk Write Queue Length	指写入请求(为所选磁盘在实例间隔中列队的)的平均数。	
PhysicalDisk	Average Disk sec/Read	指以秒计算的在此盘上读取数据的所需平均时间。	
PhysicalDisk	Average Disk sec/Transfer	指以秒计算的在此盘上写入数据的所需平均时间。	
PhysicalDisk	Disk Reads/sec	指在此盘上读取操作的速率。	
PhysicalDisk	Disk Writes/sec	指在此盘上写入操作的速率。	

判断磁盘瓶颈的方法是通过以下公式来计算：

每磁盘的 I/O 数 = [读次数 + （4 * 写次数）] / 磁盘个数

如果计算出的每磁盘的 I/O 数 大于 磁盘的处理能力，那么磁盘存在瓶颈。

7.5 Web 应用程序

这里以 ASP.NET 开发的 Web 应用程序为例进行说明。

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
ASP.NET Applications	Request/Sec Request Executing	每秒执行的请求数。 当前执行的请求数。	如果 Request/Sec 的值比较小，你的 Web 程序可能是瓶颈
ASP.NET	Request Wait Time Request Executing Time Request Queued	最近的请求在队列中等待的毫秒数。 执行最近的请求所用的毫秒数。 等候处理的请求数。该计数器应保持接近 0。超过 IIS 队列长度会出现“服务器太忙”错误。	Request Wait Time 和 Request Queued 在理想状况下应该接近 0，如果这两个值太大，那么需要重写代码提高性能

7.6 IIS5.0

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
Web Service	Bytes Sent/Sec	“送出字节数/秒”是 Web 服务送出数据字节的比率。	
Web Service	Bytes Received/Sec	“接收字节数/秒”是 Web 服务接收数据字节的比率。	

Web Service	Get Request/Sec	使用 Get 方法的 HTTP 请求速率。Get 请求用于基本文件获取或图象地图，它们可和表格一起使用。	
Web Service	POST Request/Sec	使用 Post 方法的 HTTP 请求速率。 Post 请求用于表格或网关请求。	

7.7 SQL Server

这里针对 SQL Server2000，而且只是列出比较关键的几个。更加详细的信息可以参考 SQL Server 的联机文档。

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
Processor	%Processor Time	CPU 使用率	
SQL Server: General Statistics	Logins/sec	这是每秒登录到 SQL Server 的计数。	
SQLServer:Cache Manager	Cache Hit Ratio (all instances)	显示在高速缓存中找到数据的命中率。如果数值持续小于 85%，则表示内存有问题。	
SQL Server: General Statistics	User Connections	显示当前 SQL 用户数。与 Active Server Pages：Requests/Sec 计数器进行比较，可帮助了解脚本对 SQL Server 的影响程度。如果差别过大，则表示测试脚本不能有效地对 SQL Server 进行应力测试。	
SQLServer:Locks	Lock Waits/sec	显示在当前进程完成之前强制其他进程等待的每秒锁定请求的数量。如果该值始终大于 0，则表示事务有问题。	
SQL Server: Buffer Manager	Buffer Cache Hit Ratio	计数器值依应用程序而定，但比率最好为 90% 或更高。增加内存直到这一数值持续高于 90%，表示 90% 以上的数据请求可以从数据缓冲区中获得所需数据。	
SQL Server: SQL Statistics	Batch Requests/sec	每秒收到的 Transact-SQL 命令批数。这一统计信息受所有约束（如 I/O、用户数、高速缓存大小、请求	

		的复杂程度等）影响。批请求数值高意味着吞吐量很好。	
SQL Server: Buffer Manager	Lazy Writes/sec	每秒被缓冲区管理器的惰性写入器写入的缓冲区数。惰性写入器是一个系统进程，其主要任务是刷新成批的老化的脏缓冲区（指包含更改的缓冲区，这些更改必须写回磁盘，才能使该缓冲区由其它页重新使用），并使之可由用户进程使用。 惰性写入器消除了为创建可用缓冲区而频繁执行检查点的需要。	
SQL Server: Buffer Manager	Page Reads/sec	每秒发出的物理数据库页读取数。这一统计信息显示的是在所有数据库间的物理页读取总数。由于物理 I/O 的开销大，可以通过使用更大的数据高速缓存、智能索引、更高效的查询或者改变数据库设计等方法，使开销减到最小。	
SQL Server:Databases	Transactions/sec	每秒为数据库启动的事务数。	

7.8 Network Delay

如果要监视的两台计算机在同一个局域网络内，建议不要使用 Network Delay Monitor。因为在同一局域网内，Network Delay 会非常的小，网络监视器会有足够的时间在每秒钟内发送成百上千的请求，这样会导致源计算机（source machine）的 CPU 和内存超负荷工作。

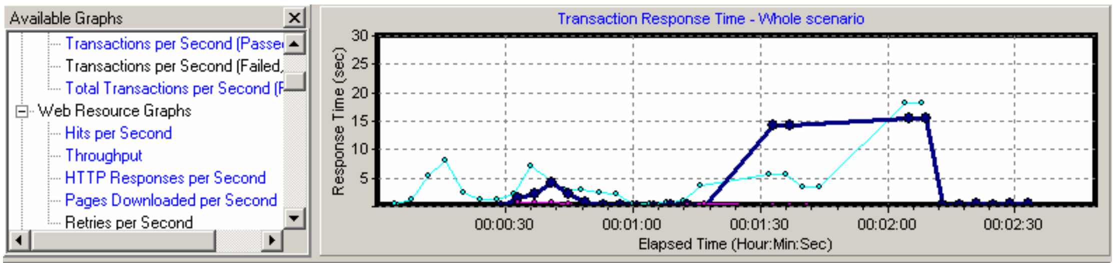
默认情况下"Enable display of network nodes by DNS names"选择是没有选中的，因为选中它会明显的降低该监视器的速度。

8 分析实时监视图表

这一章仅仅介绍几个最重要的图表。

Q1 事务响应时间是否在可接受的时间内？哪个事务用的时间最长？

看 Transaction Response Time 图，可以判断每个事务完成用的时间，从而可以判断出那个事务用的时间最长，那些事务用的时间超出预定的可接受时间。
下图可以看出，随着用户数的不断增加，login 事务的响应时间增长的最快！

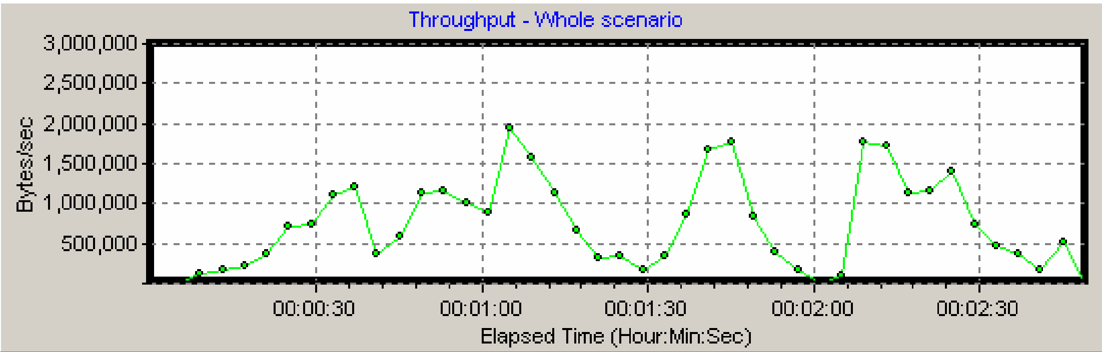


Color	Scale	Transaction	Max	Min	Avg	Std	Last
	1	Checkout1	3.886	0.010	0.118	0.290	0.024
	1	Login	19.718	0.040	4.217	6.429	0.539
	1	Default	26.508	0.030	5.055	6.283	18.221

Q2 网络带宽是否足够?

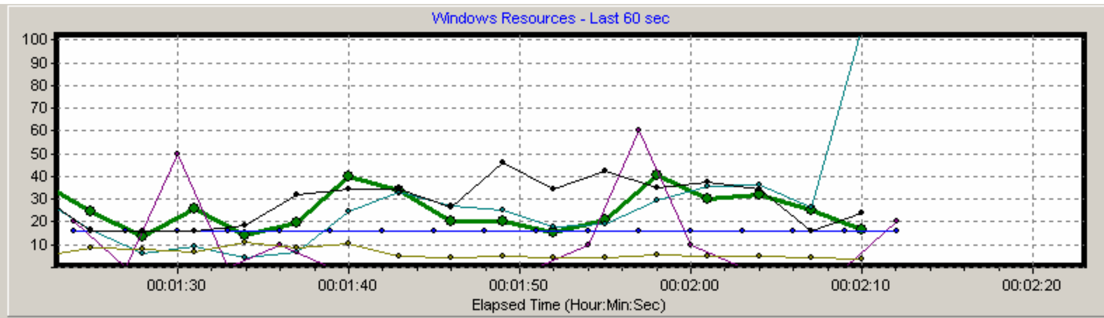
“Throughput”图显示在场景运行期间的每一秒钟，从 Web Server 上接受到的数据量的值。拿这个值和网络带宽比较，可以确定目前的网络带宽是否是瓶颈。

如果该图的曲线随着用户数的增加，没有随着增加，而是呈比较平的直线，说明目前的网络速度不能够满足目前的系统流量。



Q3 硬件和操作系统能否处理高负载?

“Windows Resources”图实时地显示了 Web Server 系统资源的使用情况。利用该图提供的数据，可以把瓶颈定位到特定机器的某个部件。



Color	Scale	Measurement	Machine	Max	Min	Avg	Std	Last
	0.1	File Data Operations/sec (System)	192.168.6...	1050.853	41.637	266.285	220.278	1050.8
	1	% Processor Time (Processor _Total)	192.168.6...	40.674	13.731	25.368	8.624	16.667

9 利用 Analysis 分析结果

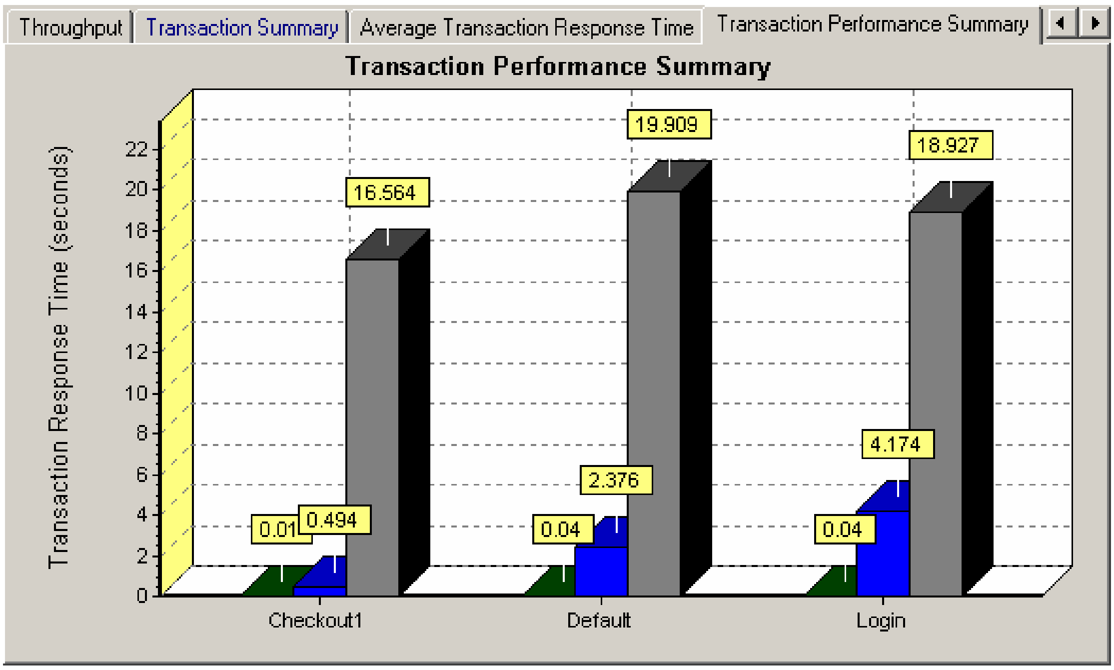
场景运行结束后，需要使用 Analysis 组件分析结果。Analysis 组件可以在“开始程序”菜单中启动，也可以在 Controller 中启动。

由于我本人对怎样分析结果最有效没有进行比较多的实践，所以这里只能按照常规的方法进行简单介绍。

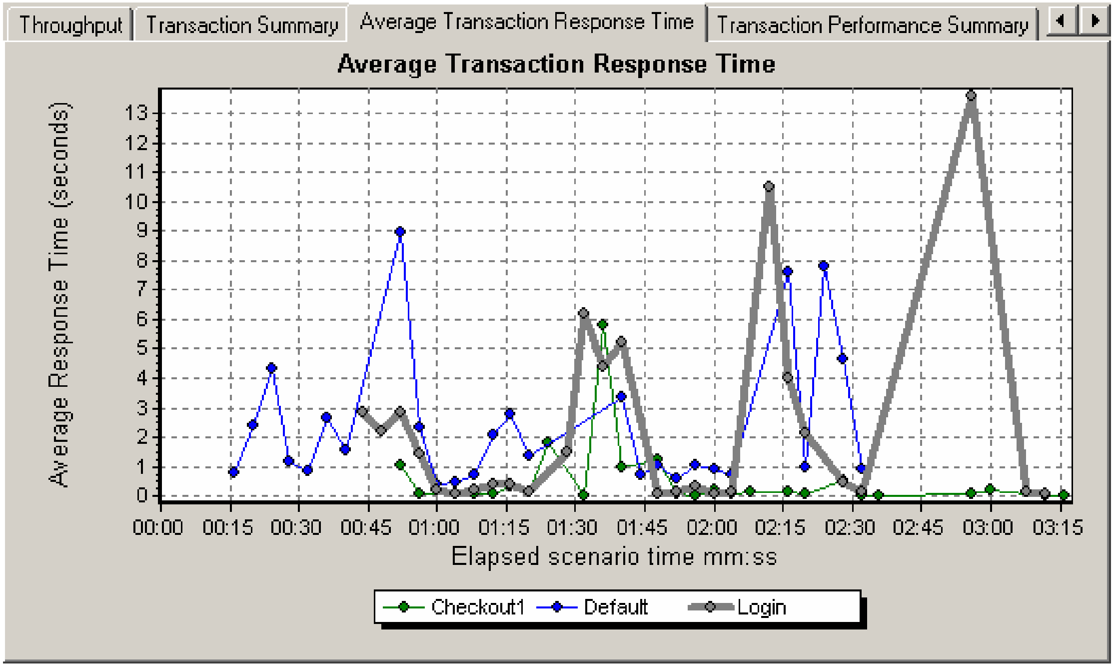
注意：这里介绍的分析方法只适用于 Web 测试。

9.1 分析事务的响应时间

第一步，看“Transaction Performance Summary”图，确认那个事务的响应时间比较大，超出了我们的标准。看下图，login 事务的平均响应时间最长。



然后我们再看“Average Transaction Response Time”，观察 login 在整个场景运行中每一秒的情况。从图中可以看出，login 事务的响应时间并不是一直都比较高，只是随着用户数的增加，响应时间才明显增加的。



为了定位问题，明白为什么 login 事务的响应时间比较长，现在我们要分解 login 事务，分析该页面上每一个元素的性能。在上图中，选择要分解的事务曲线，然后点鼠标右键，选择“ Web Page Breakdown for login ”

9.2 分解页面

通过分解页面可以得到：比较大的响应时间到底是页面的哪个组件引起的？问题出在服务器上还是网络传输上。

这里为了解说各个时间（比如：DNS 解析时间、连接时间、接受时间等）

DNS Resolution　Connection　SSL Handshaking　FTP Authentication　First Buffer　Receive　Client　Error

下面简单说一下浏览器从发送一个请求到最后显示的全过程。

1. 浏览器向 Web Server 发送请求，一般情况下，该请求首先发送到 DNS Server 把 DNS 名字解析成 IP 地址。解析的过程的时间就是 DNS Resolution。这个度量时间可以确定 DNS 服务器或者 DNS 服务器的配置是否有问题。如果 DNS Server 运行情况比较好，该值会比较小。
2. 解析出 Web Server 的 IP 地址后，请求被送到了 Web Server，然后浏览器和 Web Server 之间需要建立一个初始化连接，建立该连接的过程就是 Connection。这个度量时间可以简单的判断网络情况，也可以判断 Web Server 是否能够响应这个请求。如果正常，该值会比较小。
3. 建立连接后，从 Web Server 发出第一个数据包，经过网络传输到客户端，浏览器成功接受到第一字节的时间就是 First Buffer。这个度量时间不仅可以表示 Web Server 的延迟时间，还可以表示出网络的反应时间。

4. 从浏览器接受到第一个字节起，直到成功收到最后一个字节，下载完成止，这段时间就是

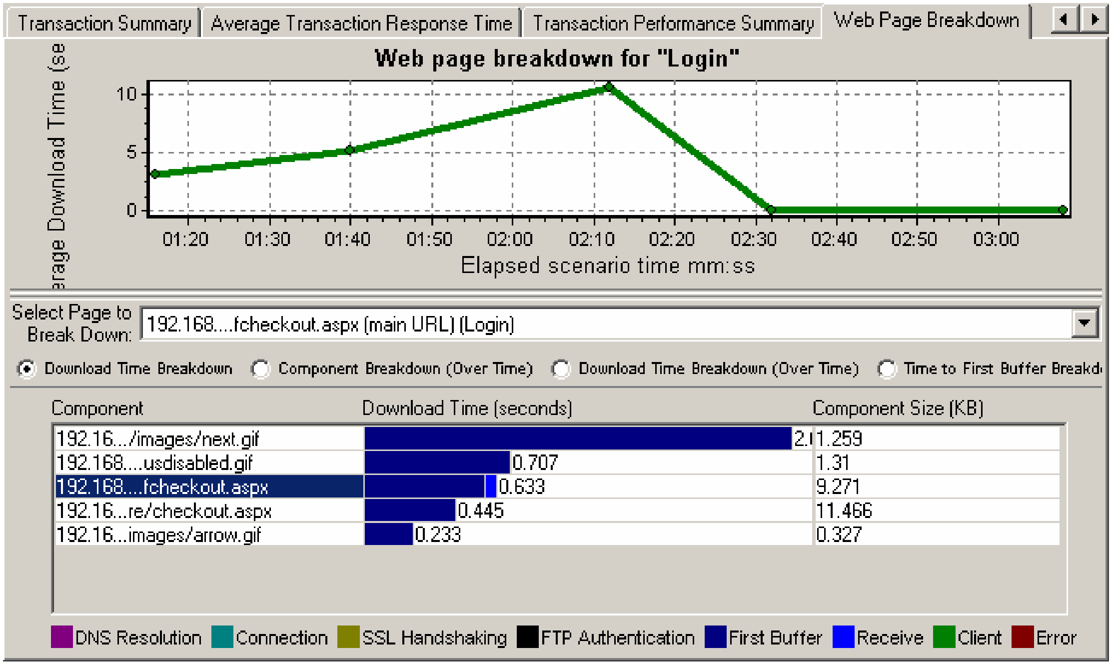
。这个度量时间可以判断网络的质量（可以用 size/time 比来计算接受速率）

其他的时间还有 SSL Handshaking（SSL 握手协议，用到该协议的页面比较少）、Client Time（请求在客户端浏览器延迟的时间，可能是由于客户端浏览器的 think time 或者客户端其他方面引起的延迟）、Error Time（从发送了一个 HTTP 请求，到 Web Server 发送回一个 HTTP 错误信息，需要的时间）。

熟悉了以上各个时间的含义后，我们开始看分解页面的图形。

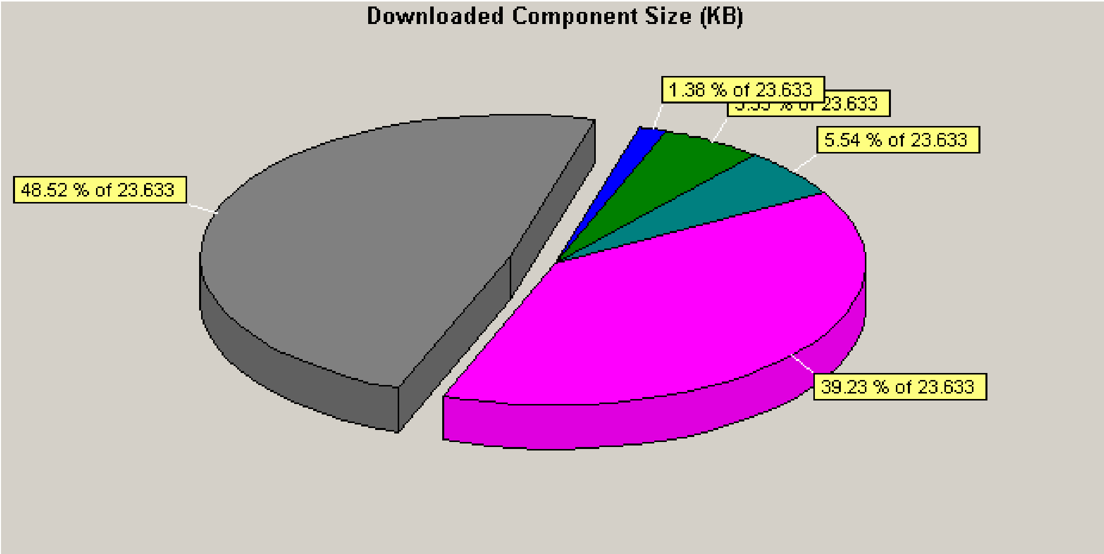
首先看“Downlaod Time Breakdown”，可以看出 login 事务分解的各个组件的大小，以及各个组件的下载时间。

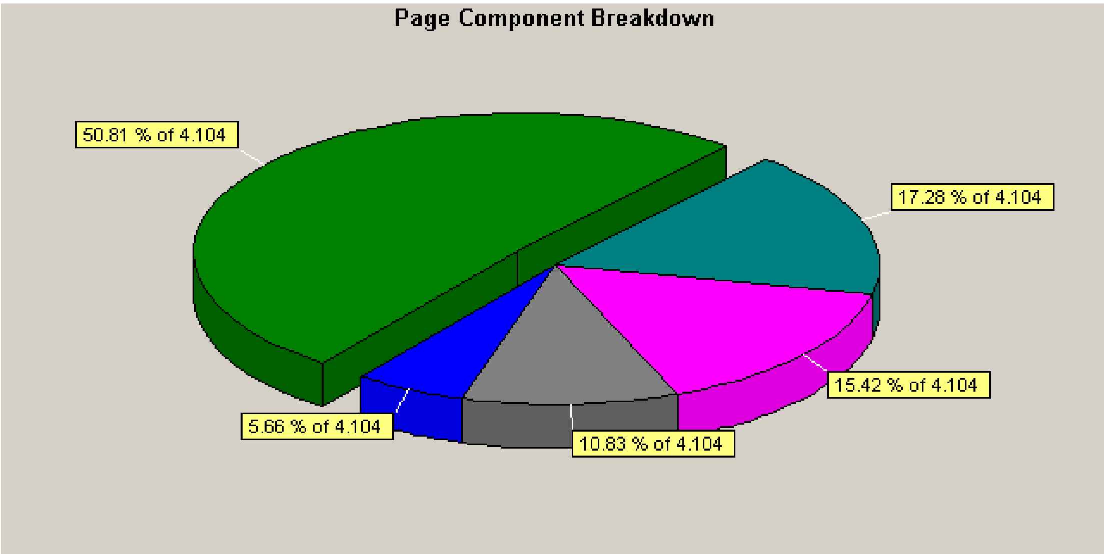
从下图可以看出，login 页面有 5 个组件组成，其中 next.gif 下载用的时间最长，并且几乎所有的时间都用在了 First Buffer 上，而其大小为 1.256KB，并不是很大。



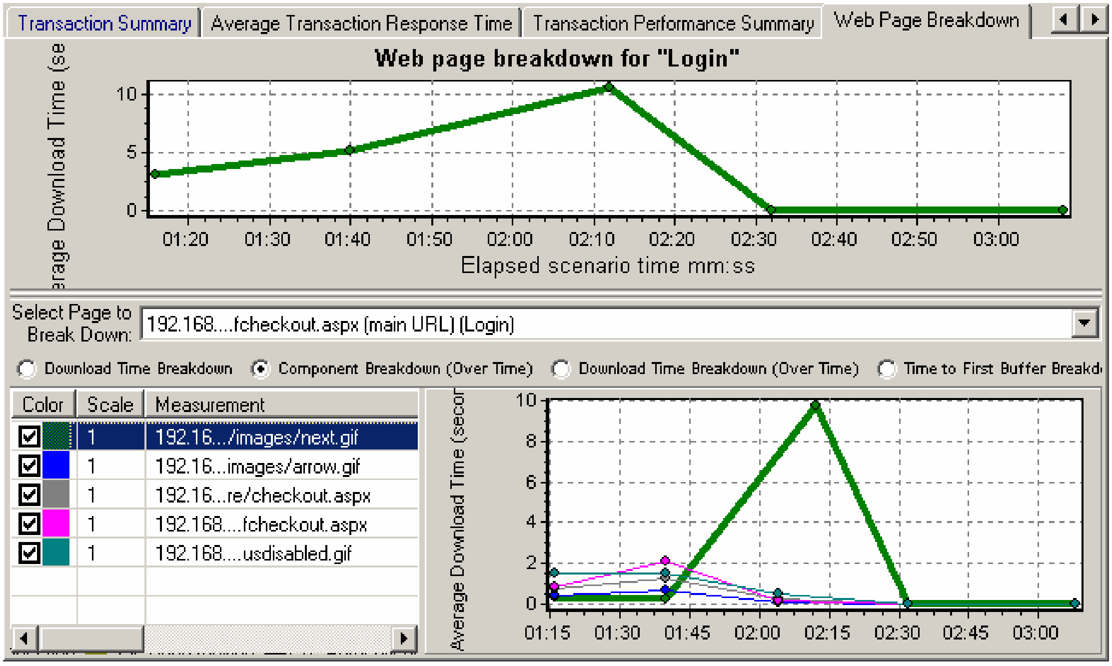
上图提供的组件大小的信息比较简单，更详细的信息参考“Download Component Size Graph”。利用该图可以看出该页面各种组件的大小所占的比例关系。

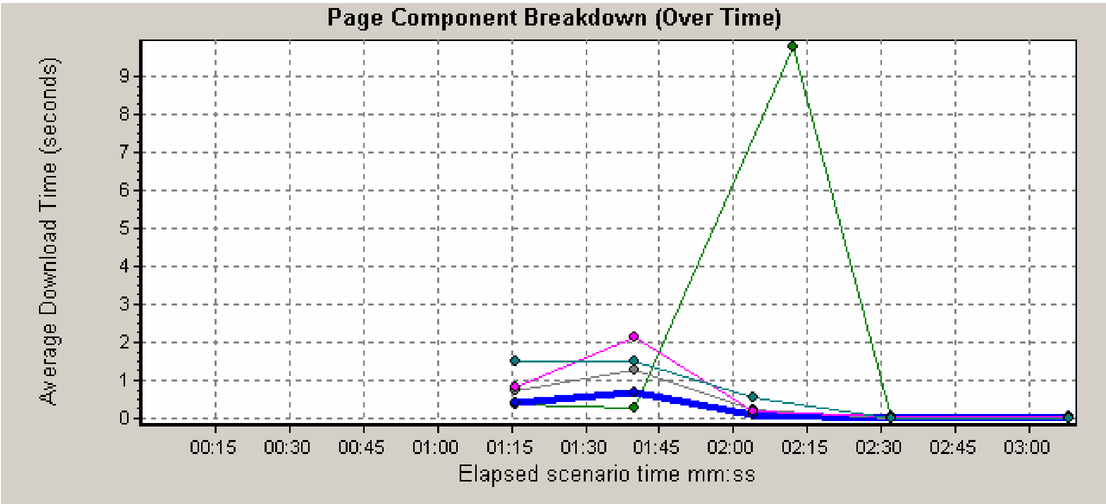
同样，要看各个组件下载时间的更详细的信息，可以参考“Page Component Breakdown”。利用该图可以看出该页面各种组件下载时间的比例关系。



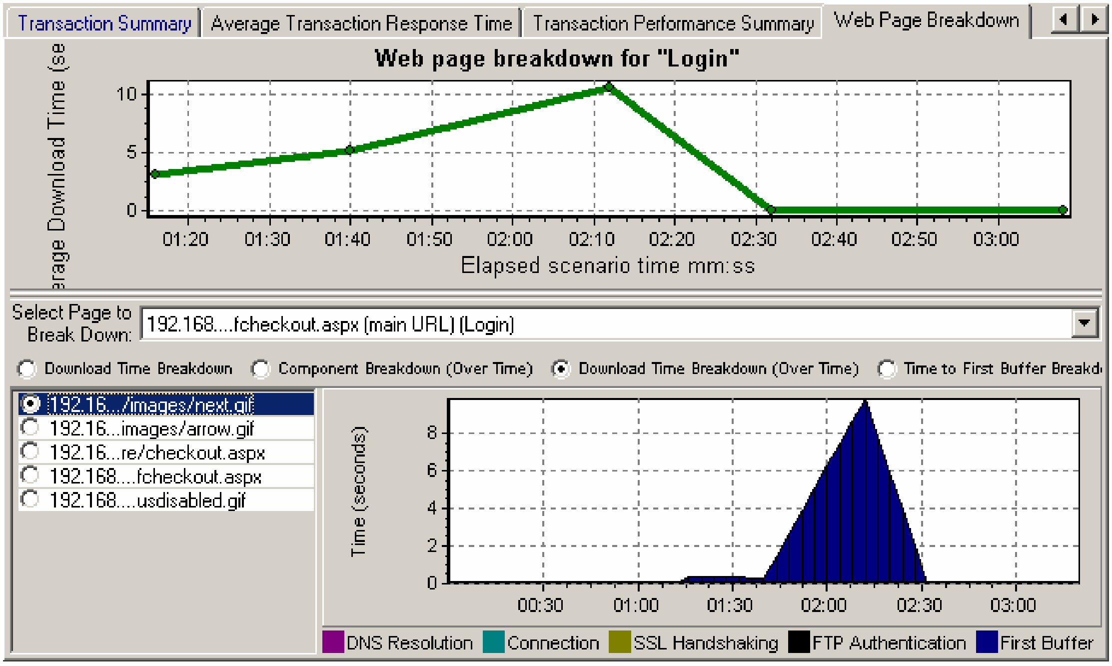


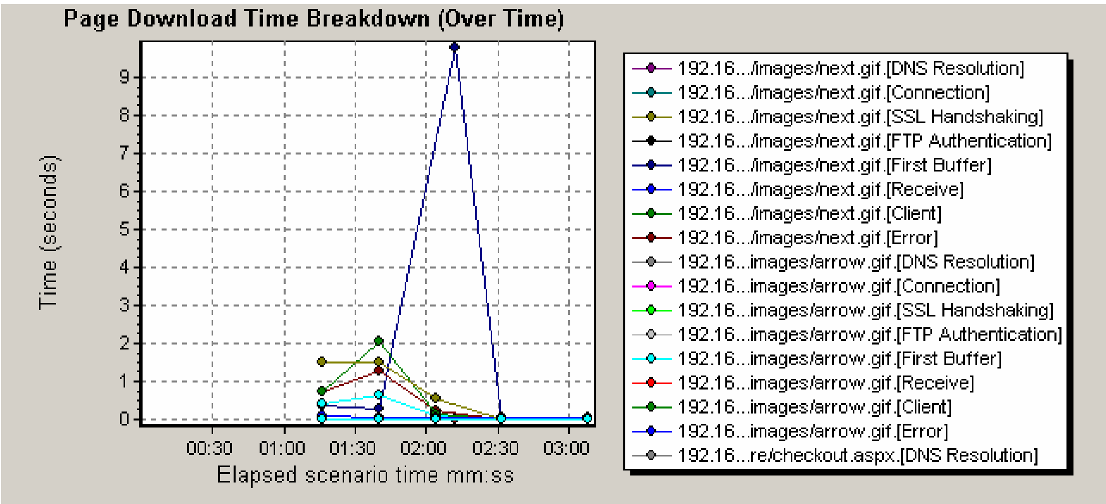
选择“ Component Breakdown（Over Time）**”，可以以图形曲线的形式看出各个组件在场景运行中的每秒钟的下载时间，比较具体。要看到具体的值，可以参考** Page Component Breakdown（Over Time）



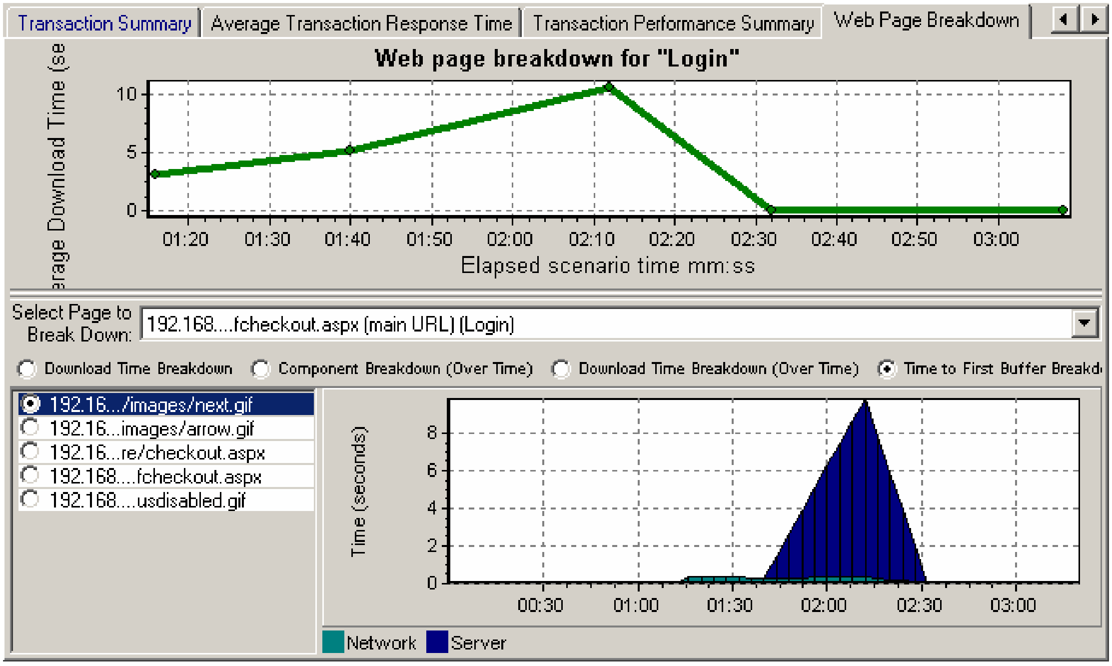


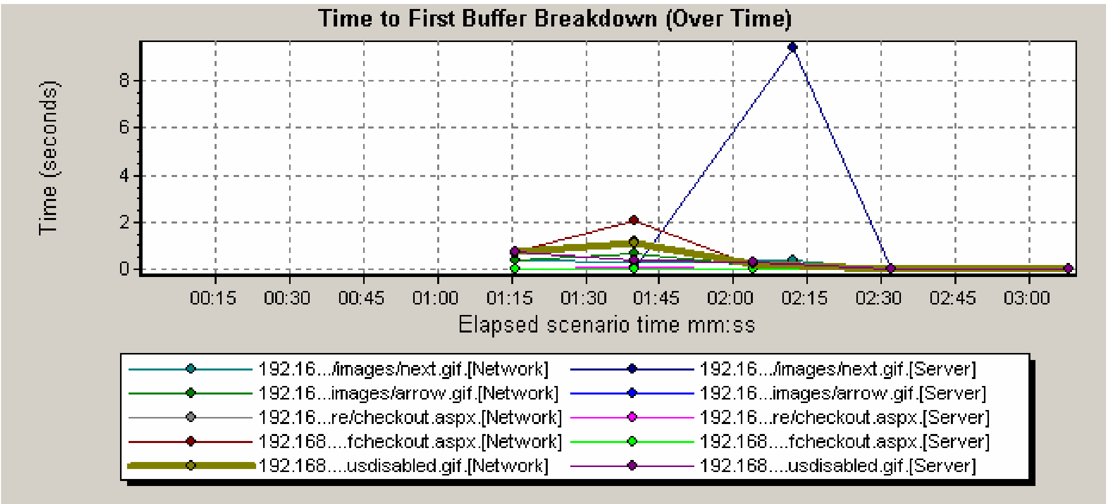
然后再选择“ Download Time BreakDown（Over Time）”，从中可以看出在场景运行中的每一秒钟，组件用在传输的各部分的时间。要看到具体的值，可以参考 Page Download Time Breakdown（Over Time）





为了确认问题缘由到底是服务器还是网络，选择“Time to First Buffer Breakdown”，可以看出 Server 时间比 network 时间要高的多，从而确定问题是服务器引起的。然后我们就可以参考 Web Server 的相关图表，来确定问题是由服务器的哪个部分引起。遵从这样的步骤，可以一步步的接近问题源；如果问题由网络引起，可以参考 NetWork 相关的图表，确定什么样的网络问题是性能的瓶颈。同时可以参考“Time To First Buffer BreakDown(Over Time)”





9.3 确定 WebServer 的问题

网站的性能问题可能是由多种因素引起的,其中大约有一半的性能问题最终归结到 Web Server、Web 应用程序和数据库服务器上。采用编程语言（ASP、JSP、ASP.NET 等）的网站非常依赖于数据库操作，这些都可能是引起性能问题的因素。

最常见的数据库问题是效率比较低的索引设计，数据碎片太多，过时的统计表以及不完善的应用程序设计。

在 20%的压力测试中，发现 Web Server 和 Web 应用程序是性能的瓶颈。这些瓶颈主要是由于服务器配置不当和资源不足。比如，编程比较差的代码以及形成的 DLL 能够使用所有的计算机处理器资源，导致了 CPU 的瓶颈。同样，对内存的操作不当和管理不善也很容易造成内存的瓶颈，所以我们建议在排除其他可能的因素外，首先检查 CPU 和物理内存。

9.4 其他有用的功能

9.4.1 比较每次运行的结果

一般情况下，我们进行性能测试的步骤是这样：

1 **首先进行一次性能测试，记录下结果，然后分析结果，提出改进的建议**

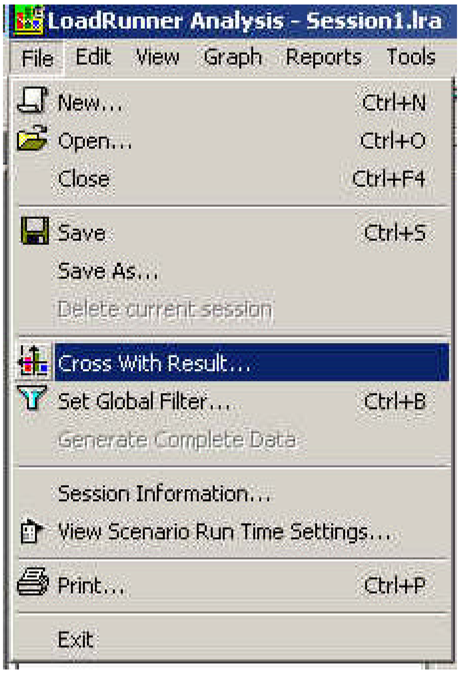
2 **开发人员根据建议对代码或者服务器的配置进行修改**

3 **测试人员在相同的条件下进行第二轮测试**

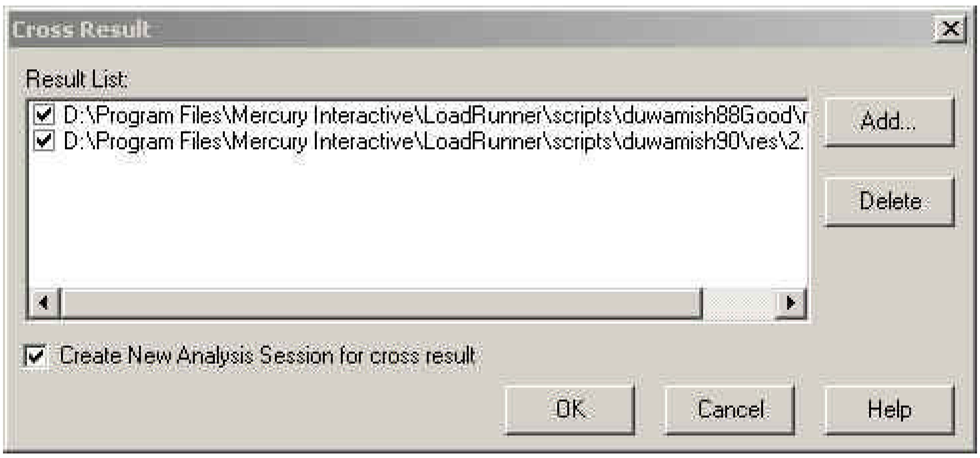
4 **测试人员对两轮测试结果比较，确定开发人员修改的结果是否有效**

那么在 Analysis **中怎样进行对两轮结果进行比较呢？**

可以采用菜单操作：



然后出现对话框，需要选择需要比较的结果文件(lrr),选择多个



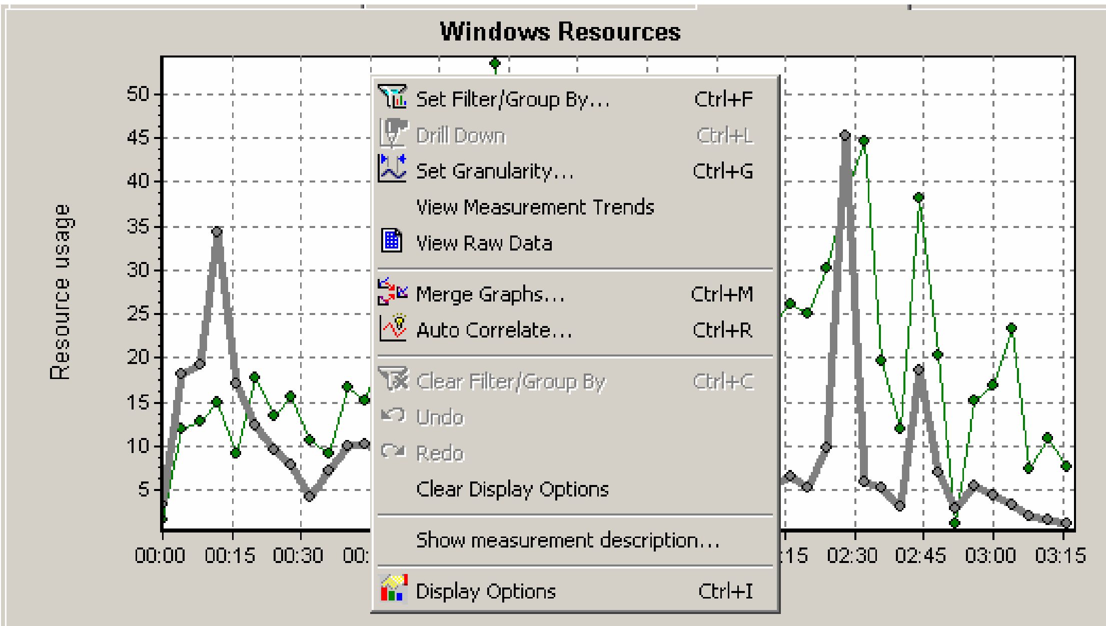
OK 即可。

9.4.2 对图表进行组合合并

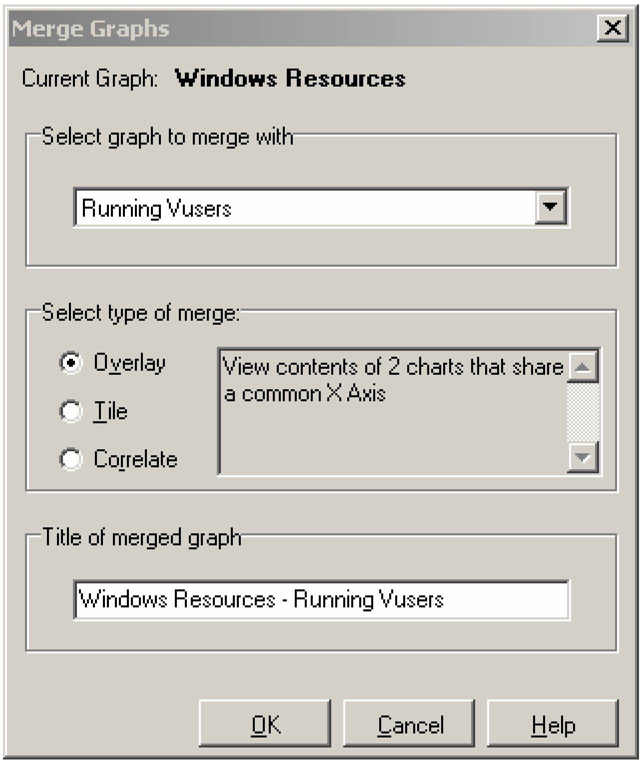
Analysis 默认的图表都是以时间作为横坐标，然而在分析结果的过程中，我们可能需要以“运行的用户数”作为横坐标，来比较结果。

假如我们要画出 Windows Resources——VUsers 的图表，可以这样操作。

首先打开 Windows Resources 图表，然后在图表上点鼠标右键，选择 Merge Graphs

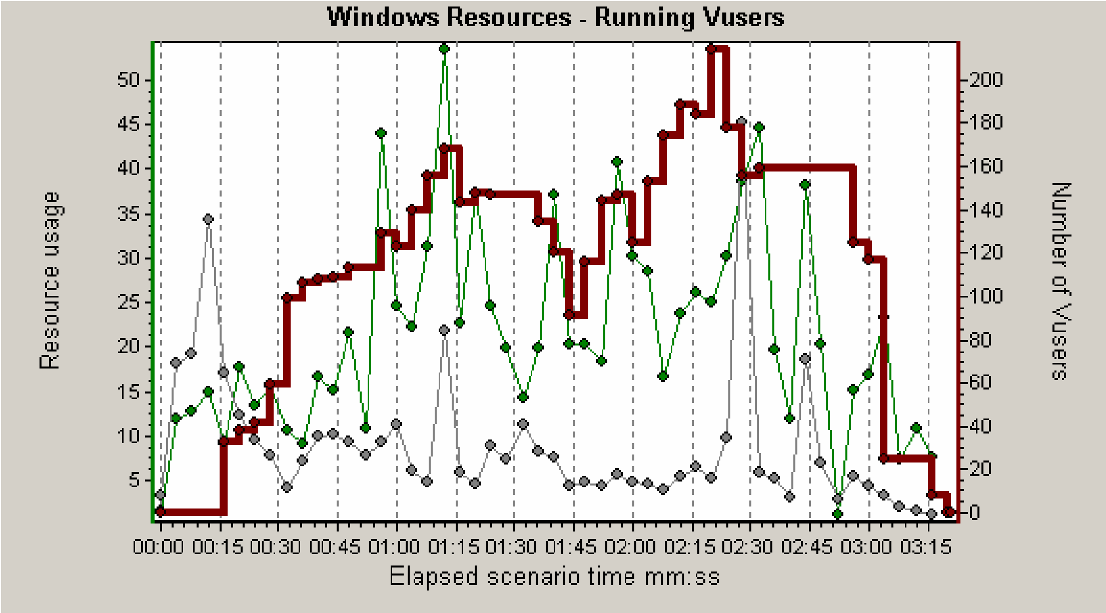


出现 Merge Graphs 对话框

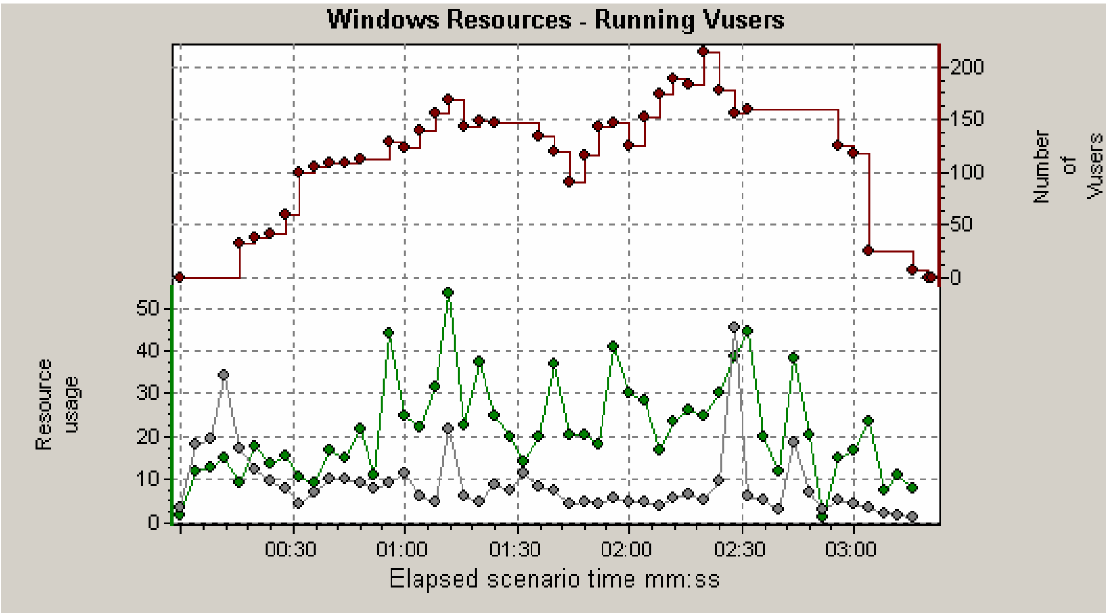


选择第一项“Overlay”，出现以下的图表，这样是把两个图表进行了合并，两条曲线的纵轴

共用一个原点，横轴还是时间轴。



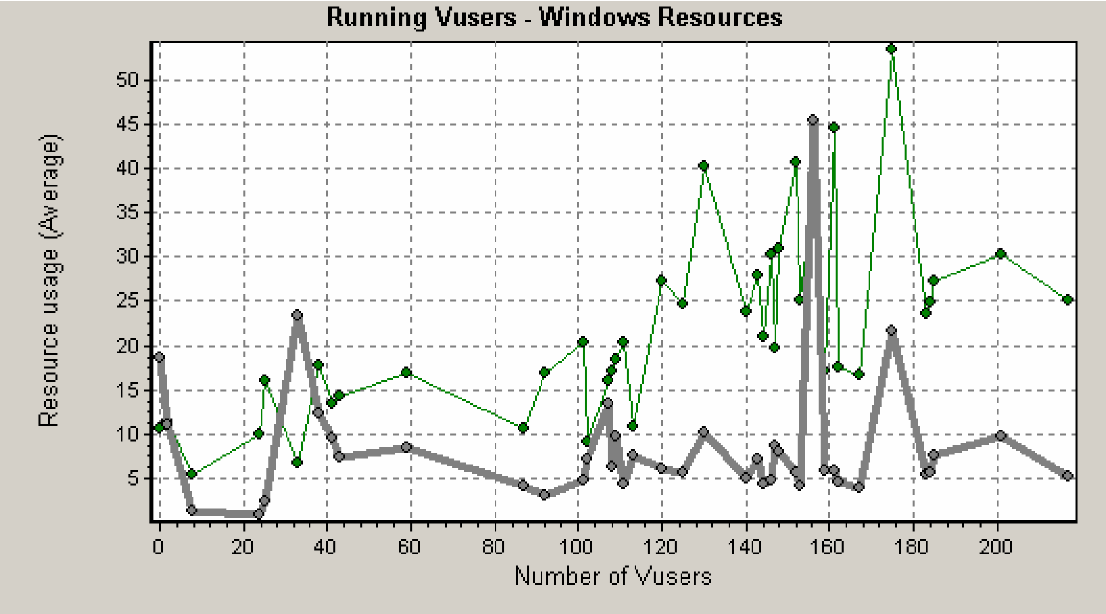
选择第二项“ Title ”，出现以下的图表，这样是把两个图表进行了合并，两条曲线的纵轴不再共用一个原点，VUsers 的原点在 Windows Resouces 的上面，横轴还是时间轴。



选择第三项“ Correlate ”，LoadRunner 提示信息



我们应该这样，打开 Running VUsers 图表，在图表上执行前面提到的操作，不过选择第三项“ Correlate ”，即可，显示图表如下图



这样就是以运行的用户数为横轴，Windows Resouces 为纵轴画出的图。

10 经常遇到的问题

10.1 VuGen 的问题

在使用 VuGen 中经常会遇到的问题。

10.2 Controller 的问题

在使用 Controller 中经常会遇到的问题。

1. 在添加完 Load Generators 机器时，连接老是失败；添加的机器明明已经安装了 loadrunner，并且网络通讯正常。
   解决方法：在安装 loadrunner 的第七步骤，应该选择第 2 项，如果选择了第一项，就会有这种问题。重新安装一下即可。
2. 在 VuGen 中运行良好的脚本，到 Controller 中运行却出问题。
   这种问题可能会遇到。为了确定问题出在 Controller 中的场景，而不是脚本的问题，你应该在所有的 Load Generators 机器上使用 VuGen 运行测试脚本，确保都能够运行正确。因为 VuGen 和 Controller 运行的机制不一样。在 VuGen 中运行时使用的是完整的浏览器，而在 Controller 中运行时使用的只是浏览器的基本的部分。

10.3 计数器的问题

在使用性能计数器中经常会遇到的问题。

1. 添加了 Windows Resources 计数器后，却看不到实时的数据。

解决方法：要得到监视的数据，必须要在被监视的服务器（Web Server）上获得管理员权限。最简单的方法是在“网络邻居”中以 administrator 身份登陆 Web Server。当然使用下面的控制台命令也可以：net use \\<机器名> 然后登陆用户名和密码即可。（登陆的用户名必须具有管理员权限）

2. **添加了一些默认的性能计数器后，出现了错误。**
   解决方法：可能是一些 LoadRunner 默认的计数器在 WebServer 上已经不存在的原因，尤其是数据库的计数器方面。简单的解决方法，就是删除有问题的计数器，添加比较接近的计数器（可能需要参考 Windows 帮助或者数据库的帮助）